MEDAS : Bases de données et Systèmes d'informations
XML

Raphaël Fournier-S'niehotta

Équipe Vertigo
Laboratoire CEDRIC
Conservatoire National des Arts & Métiers, Paris, France

le cnam

Plan du cours

Plan du cours

Historique

1969 GML : Langage permettant de dissocier contenu des spécificités techniques des imprimantes ;

1986 SGML (ISO, organisation internationale de normalisation) : Standard Generalized Markup Language. Successeur de GML.

1991 HTML : Application de SGML pour le web. "Mauvaise" utilisation de SGML.

Historique (suite)

La gestion des données du web a tout d'abord été fondée sur HTML, qui contient à la fois contenu et présentation "mélangés"

- HTML est approprié pour les humains : permet des sorties sophistiquées et les interactions avec texte et images,
- HTML devient limité lorsque il s'agit d'exploiter automatiquement les données par les logiciels.

Nécessité d'un format de publication dissociant contenu et présentation, de façon à rendre le contenu exploitable par un programme.

Historique (suite)

Les données (éventuellement volumineuses) circulent :

- en interne dans l'entreprise,
- vers l'extérieur.

entre plate-formes d'exploitation éventuellement différentes.

Nécessité d'un format d'échange de contenu "souple" s'auto-décrivant (encodage et contenu) indépendant de toute plate-forme.

Historique (suite)

En repartant de SGML, création de XML qui est une application de SGML pour le web mais pas que !
Validé par le W3C (World Wide Web Consortium).

Historique (suite)

XML décrit du contenu, facilite la communication machine-à-machine et l'échange des données

- XML est un format de données générique permettant de manipuler données structurées et semi-structurées,
- XML facilite l'intégration des données à partir du moment où source et destination partagent maintenant un language commun,
- XML vient avec toute une panoplie de logiciels, APIs, outils, applications.

Plan du cours

Un modèle pour les données semi-structurées

Idées de base :

- Modèle de données, fondé sur les arbres (plus généralement les graphes), permettant de représenter des données régulières et des données irrégulières (mais pas trop). → **données semi-structurées**.
- Typage souple.
- Représentable en *serialized form* afin d'être **échangé entre applications**.

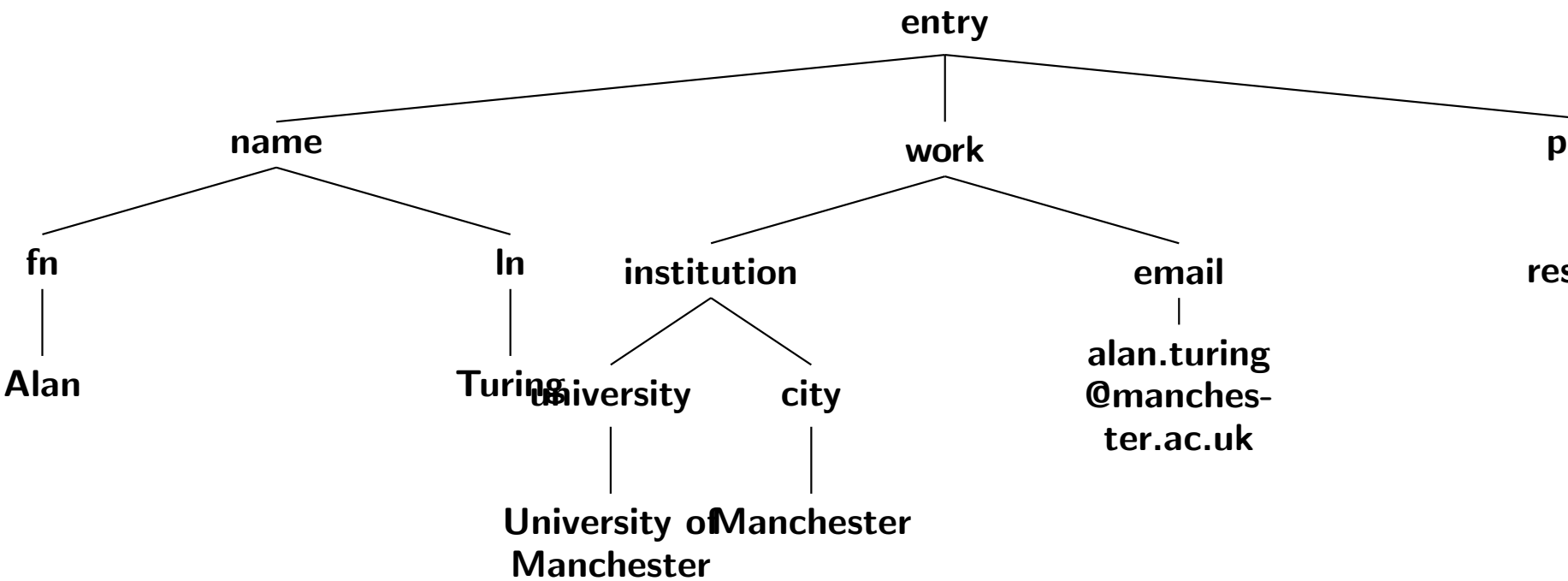
Document XML

Un document XML est un arbre pour lequel :

- les nœuds sont étiquetés,
- le nombre de fils d'un nœud n'est pas limité,
- les fils d'un nœud sont ordonnés.

Document XML

Un document XML représente un arbre.

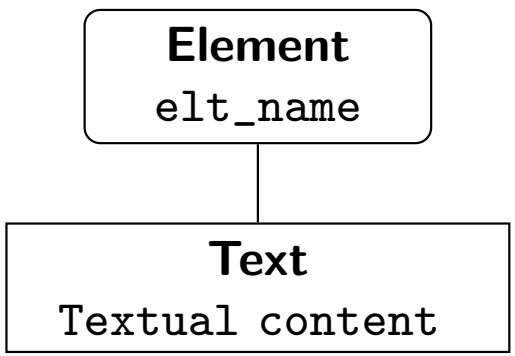


Document XML

Les composants fondamentaux d'un document XML sont l'*élément* et le *texte*.

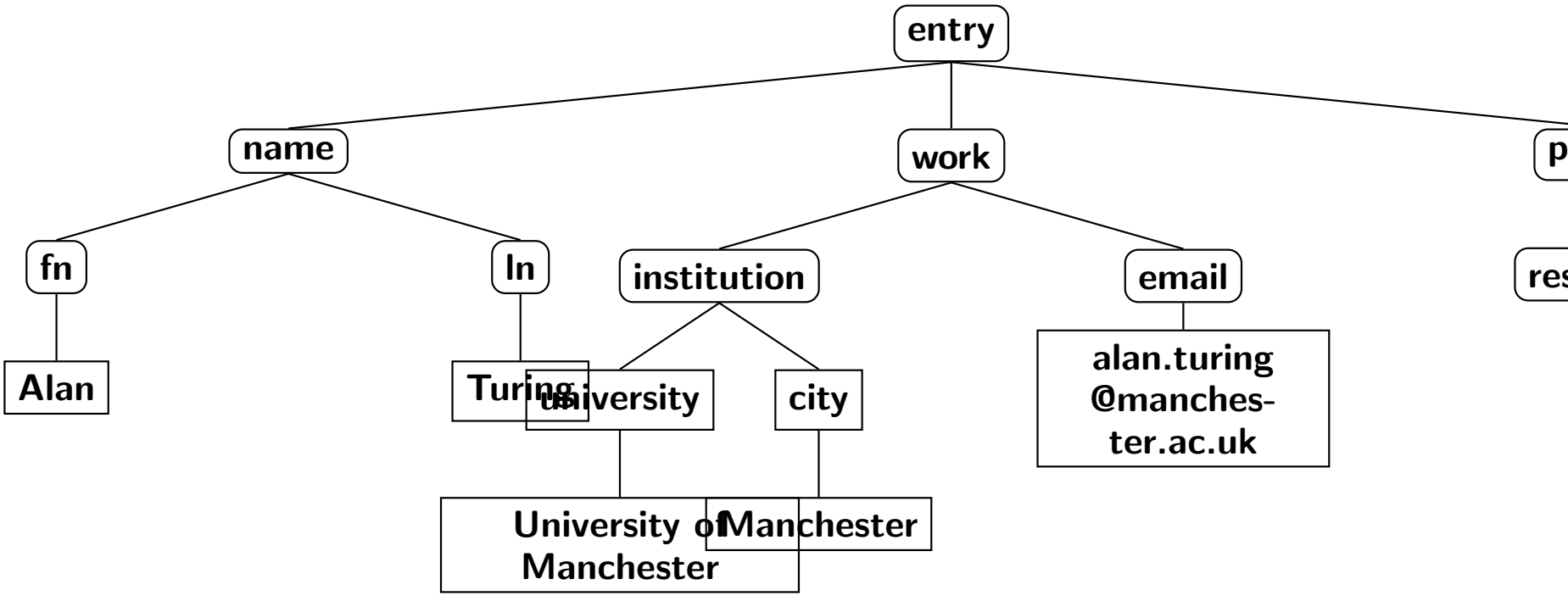
```
<elt_name>
  Textual content
</elt_name>
```

Sous forme d'arbre :



Document XML

Un document XML représente un arbre.



Forme sérialisée

Avec indentation pour améliorer la lisibilité :

```
<entry>
 <name>
  <fn>Alan</fn>
  <ln>Turing</ln>
 </name>
 <work>
  <institution>
   <university>University of Manchester<university>
   <city>Manchester</city>
  </institution>
  <email>alan.turing@manchester.ac.uk</email>
 </work>
 <purpose>researcher</purpose>
</entry>
```

Forme sérialisée

Un document peut être sérialisé (obtenu par parcours d'arbre préfixe) :

```
<entry><name><fn>Alan</fn><ln>Turing</ln></name><work><institution>
<university>University of Manchester<university><city>Manchester</city></
institution><email>alan.turing@manchester.ac.uk</email></work><purpose>
researcher</purpose></entry>
```

Mais attention !

XML définit **juste une syntaxe** : aucune sémantique n'est *a priori* associée aux étiquettes.

Document XML

Les programmes peuvent difficilement interpréter un contenu non structuré :

```
The book ‘‘Fundations of Databases’’, written by Serge Abiteboul, Rick Hull and
Victor Vianu, published in 1995 by Addison-Wesley
```

Document XML (suite)

XML fournit un moyen de structurer ce contenu :

```
<bibliography>
  <book>
    <title> Foundations of Databases </title>
    <author> Serge Abiteboul </author>
    <author> Rick Hull </author>
    <author> Victor Vianu </author>
    <publisher> Addison–Wesley </publisher>
    <year> 1995 </year>
  </book>
  <book>...</book>
</bibliography>
```

Un programme peut alors accéder à l'arbre XML, en extraire des parties, renommer des étiquettes, ré-organiser le contenu de l'arbre ou transformer l'arbre en un document ayant une structure complètement différente, etc.

Processeur XML et forme sérialisée

- La forme sérialisée est une représentation textuelle linéaire d'un arbre satisfaisant une syntaxe.
- Il existe un modèle orienté-objet permettant de représenter des données XML sous forme d'arbre : Document Object Model (W3C). Les applications "travaillent" avec ce genre de représentations sous forme d'arbre.

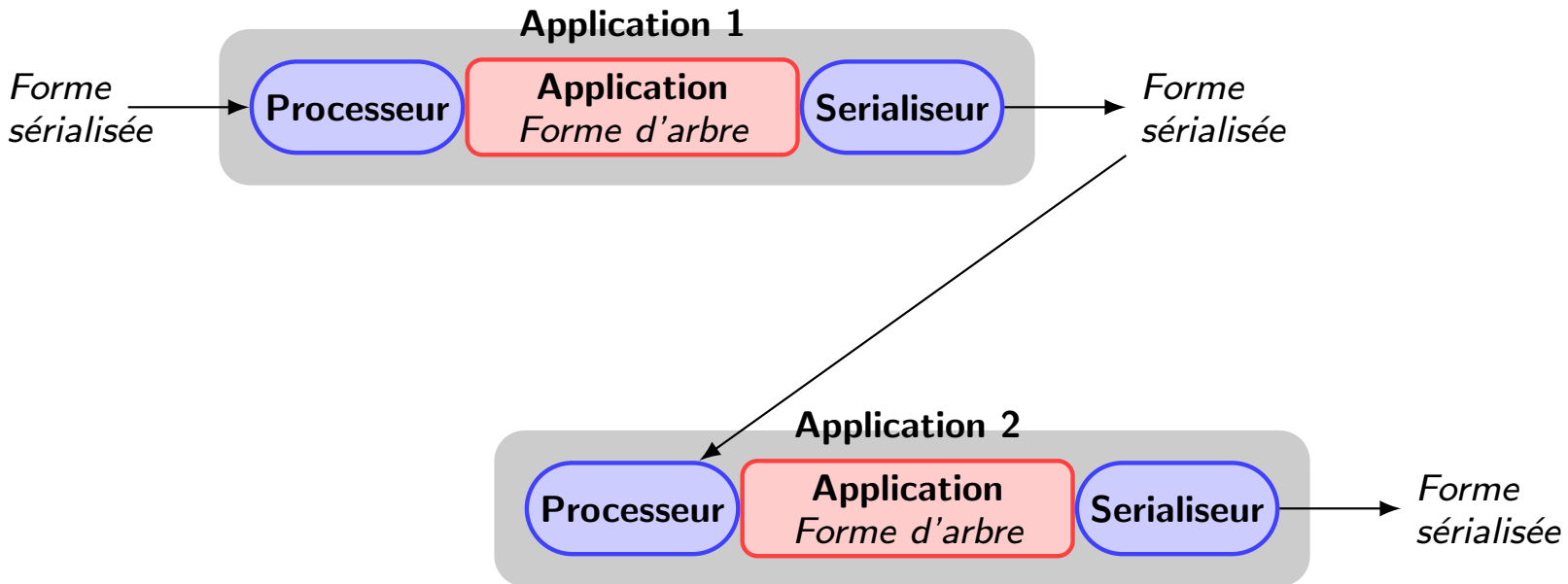
Comportement typique d'une application acceptant des documents XML en entrée :

Le formalisme XML, pourquoi ?

Format convention utilisée pour représenter des données.

Ouvert spécification publiquement accessible.

Normalisé validé par un organisme de normalisation (ici, le W3C).



Objectif

Interopérabilité des applications.

Schéma

Il est possible d'associer un schéma à un ou plusieurs documents XML.
Un schéma définit un ensemble de contraintes syntaxiques que des documents doivent satisfaire.

```
<!DOCTYPE people_list [
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>]>

<people_list>
  <person>
    <name>Martin Lodges</name>
    <birthdate>09/07/1965</birthdate>
    <gender>Male</gender>
  </person>
  <person>
    <name>Mary Logging</name>
    <birthdate>27/11/1957</birthdate>
    <gender>Female</gender>
  </person>
</people_list>
```

Toujours dans le même objectif

Interopérabilité des applications.

Et la sémantique ?

Que décrivent les documents XML suivants ?

```
<OFX>
 <SIGNONMSGSRQV1>
  <SONRQ>
   <DTCLIENT>2007101500[-8:PST]</DTCLIENT>
   <USERID>Greg123</USERID>
   <USERPASS>Greg</USERPASS>
   <LANGUAGE>ENG</LANGUAGE>
   <FI>
    <ORG>MYBANK</ORG>
    <FID>01234</FID>
   </FI>
   <APPID>QWIN</APPID>
   <APPVER>0900</APPVER>
  </SONRQ>
 </SIGNONMSGSRQV1>
 <BANKMSGSRQV1>
  <STMTTRNRQ>
...
```

```
<fiches>
 <livre titre="XML" ISBN="123456">
 <auteur> Martin Smith </auteur>
 </livre>
 <livre titre="RDF(S)" ISBN="654321" />
 <auteur> Marilyn Berton </auteur>
 </livre>
</fiches>
```

XML définit une syntaxe pour l'échange de documents

Le language XML fournit une syntaxe mais *pas de sémantique* aux documents échangés.

le cnam

Morale

Morale

Interopérabilité oui, mais à condition d'**associer une sémantique aux documents**.

Plan du cours

L"'univers" XML

- Parseurs (non validant ou validant),
- Validateurs,
- Langages de transformation,
- Langages de programmation,
- Bases de données XML,
- Éditeurs de documents XML,
- Dialectes,
- ...

Dialectes XML

Quelques applications standardisées de XML.
Dialecte XML = schéma + interprétation associée.

MathML	Description d'équations mathématiques,
XMLA	Description de données multidimentionnelles (p.e. OLAP),
XHTML	Pour l'échange et la publication de document sur le web,
XSLT	Définir des transformations à effectuer sur un document XML,
MusicXML	Description de partitions musicales (symbolique),
RSS	Syndication de contenu web,
SVG	Description de vecteurs graphiques à deux dimensions, statiques ou animés,
OFX	Description de données financières,
WML	Wireless ML utilisé dans les sites web par les applications sans-fil basées sur WAP (Wireless Application Protocol),
...	

Dialectes XML : quelques exemples

XHTML :

- syntaxe (restriction de XML) : lexique et grammaire (DTD) définissant la forme d'un document XHTML disponible à `http://www.w3.org/TR/xhtml1/#dtds`
- sémantique : l'interprétation d'un document est définie. Voir `http://www.w3.org/TR/2008/REC-xhtml-basic-20080729/`

```
<h1>Sujets d'ouverture ann&#233;e 2010-2011.</h1>

<h2>ID et IF classiques</h2>
<ul>
 <li>Grp 1 <i>Gestion de XML dans les bases de donn&#233;es relationnelles</i>. Sujet valid&#233;. </li>
 <li>Grp 2 - <i>Etude et mise en oeuvre des flux RSS</i>. Sujet valid&#233;. </li>
 <li>Grp 3 - <i>OFX</i>. Sujet valid&#233;. </li>
 <li>Grp 4 - <i>Int&#233;gration et gestion des fichiers XML sous SQLServer Integration Services 2008</i>. </li>
 <li>Grp 5 - <i>IFX - Interactive Financial eXchange</i>. Sujet valid&#233;. </li>
 <li>Grp 6 - <i>XBRL - eXtensible Business Reporting Language</i>. Sujet valid&#233;. </li>
 <li>Grp 7 - <i>Utilisation de XML dans les services Web et alternatives</i>. Sujet valid&#233;. </li>
 <li>Grp 8 - <i>Ajax - Asynchronous JavaScript and XML</i>. Sujet valid&#233;. </li>
</ul>

<h3>Planning des expos&#233;s :</h3>
```

Listing 1 – Extrait d'un document XHTML

le cnam

Dialectes XML : quelques exemples (suite)

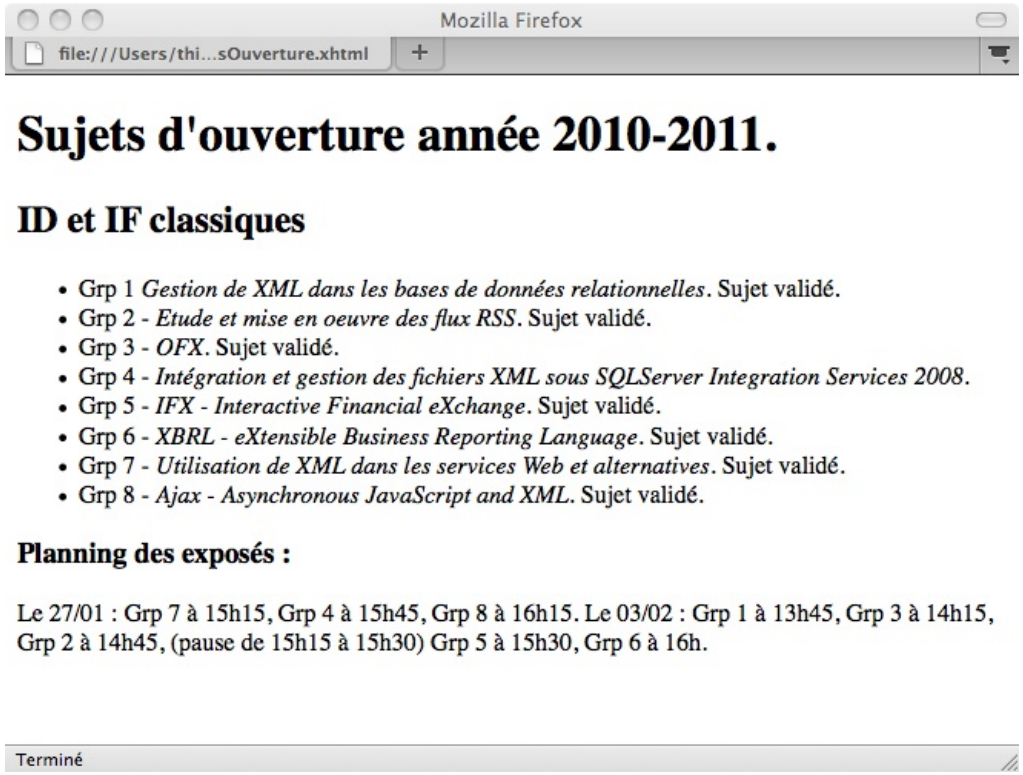


FIGURE 1 – Document *interprété* par Firefox

Dialectes XML : quelques exemples (suite)

MathML permet de décrire des formules mathématiques :

- syntaxe (restriction de XML) : DTD accessible à `http://www.w3.org/Math/DTD/mathml3/mathml3.dtd`
- sémantique : voir `http://www.w3.org/TR/MathML3`

```
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
</mrow>
```

Listing 2 – Extrait d'un document MathML

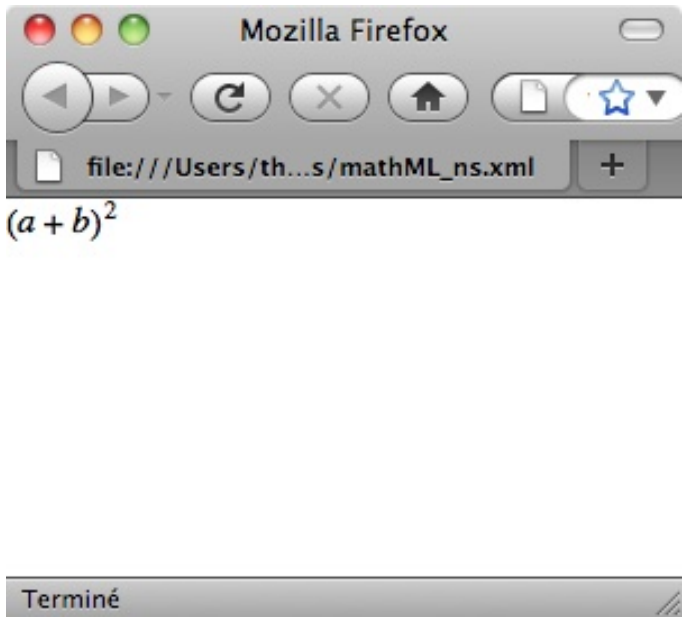


FIGURE 2 – Document *interprété* par Firefox

le cnam

Dialectes XML : quelques exemples (suite)

SVG permet de décrire des images :

- syntaxe (restriction de XML) : DTD accessible à `http://www.w3.org/TR/SVG11/svgdtd.html`
- sémantique : voir `http://www.w3.org/TR/SVG11`

```
<defs>
  <symbol id="MySymbol" viewBox="0 0 20 20">
    <desc>MySymbol – four rectangles in a grid</desc>
    <rect x="1" y="1" width="8" height="8"/>
    <rect x="11" y="1" width="8" height="8"/>
    <rect x="1" y="11" width="8" height="8"/>
    <rect x="11" y="11" width="8" height="8"/>
  </symbol>
</defs>
<rect x=".1" y=".1" width="99.8" height="29.8"
      fill="none" stroke="blue" stroke-width=".2" />
```

Listing 3 – Extrait d'un document SVG [?]

FIGURE 3 – Document *interprété* par Firefox

le cnam

Dialectes XML : quelques exemples (suite)

MusicXML permer de décrire des partitions musicales (*) :

- syntaxe (restriction de XML sous forme de DTD ou de XML schéma) : voir `http://www.recordare.com/musicxml/specification`
- sémantique : voir p.e. `http://www.recordare.com/musicxml/specification/alphabetical-index`

```
<note default-x="111">
  <pitch>
    <step>C</step>
    <octave>6</octave>
  </pitch>
  <duration>1</duration>
  <voice>1</voice>
  <type>16th</type>
  <stem default-y="50">up</stem>
  <staff>1</staff>
  <beam number="1">begin</beam>
  <beam number="2">begin</beam>
</note>
<note default-x="143">
  <pitch>
    <step>A</step>
    <octave>5</octave>
```

Listing 4 – Extrait d'un document MusicXML

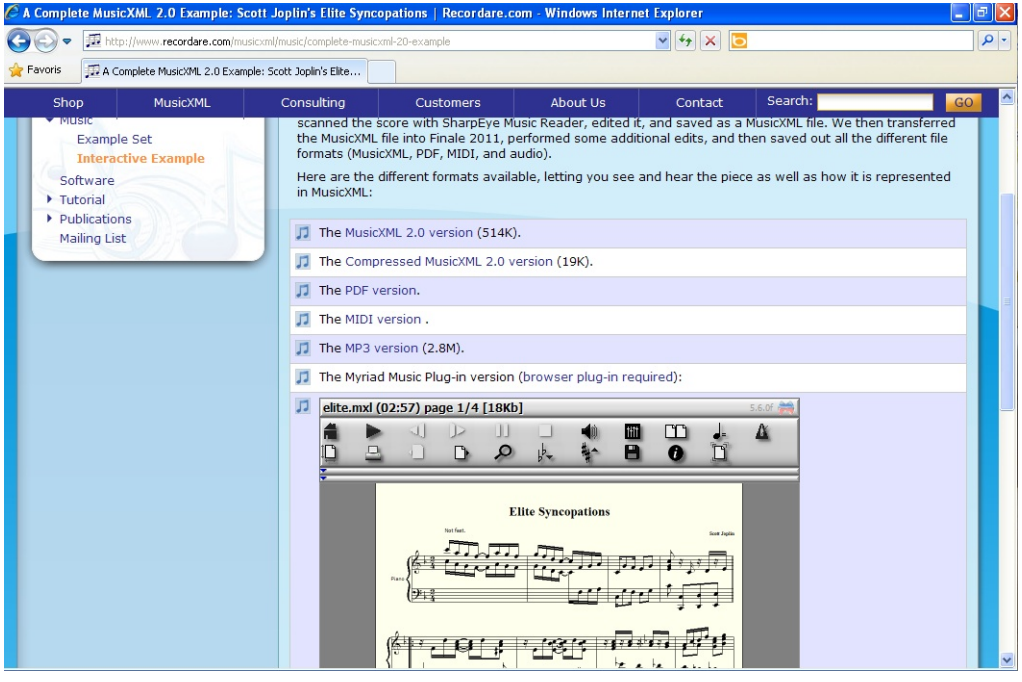


FIGURE 4 – Partition affichée sous Internet Explorer (Plug-in Myriad Music)

le cnam

Dialectes XML : quelques exemples (suite)

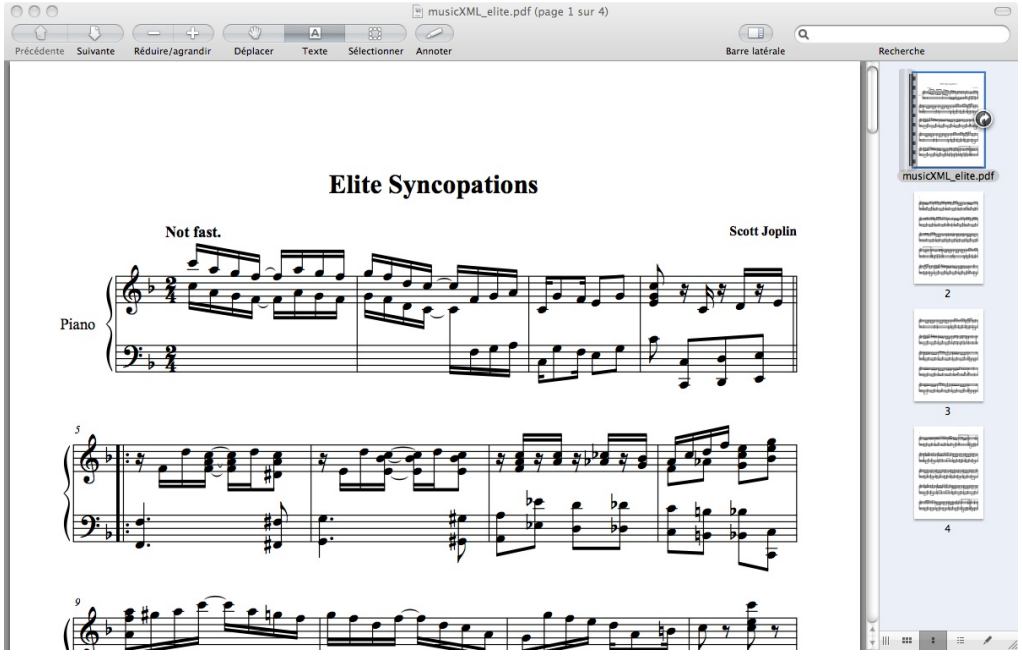


FIGURE 5 – Partition au format PDF générée par Finale (logiciel d'édition de partitions musicales) à partir du fichier MusicXML

(*) Cet exemple a été emprunté au site *Recordare.com* à

`http://www.recordare.com/musicxml/music/complete-musicxml-20-example` (en juin 2011)

Quelques standards XML

XPath the XML Path Language is a language for addressing portions of an XML document.

XQuery is a flexible query language for extracting information from collections of XML documents.

XSLT the Extensible Stylesheet Language Transformations is a language for specifying the transformation of XML documents into other XML documents.

DOM the Document Object Model, is an object model for representing (HTML and) XML document independently of the programming language.

SAX the Simple API for XML, sees an XML document as a sequence of tokens (its serialization).

Web services provide interoperability between machines based on Web protocols.

... et bien d'autres

Plan du cours

Plan du cours

Un document XML

```
<?xml version="1.0" encoding="utf-8"?>

<!-- Auteur : M. Philippe Rigaux -->

<cours code="NFE204">
        <formation name="CNAM Paris"></formation>
        <titre>
            NFE204 - Bases de donn&#233;es documentaires
        </titre>
        <notion>XML</notion>
        <notion>DTD</notion>
        <notion>XSLT</notion>
        <notion>DOX</notion>
</cours>
```

Listing 5 – Un simple fichier XML

Un document XML (suite)

Un document XML est composé de :

1. Un prologue
   1. déclaration de document XML
      `<?xml version = "1.x" [encoding = "norme" ][standalone = "yes/no" ]? >`
   2. suivie éventuellement (sans contrainte d'ordre)
      - d'une déclaration ou (exclusif) une référence à une DTD
        `<!DOCTYPE element racine [schema racine ] >` ou
        `<!DOCTYPE element racine SYSTEM "uri schema racine" >`
      - de commentaires `<!-- texte -->`
      - d'instructions de traitement (par ex. appel à XSLT)
        `<?instruction attribut1 = "valeur" ... attribut n ="valeur" ? >`
2. Un corps
   - la description de l'élément racine (le document)

```
arbre := elt_feuille | elt_noeud
elt_feuille := '<'nom_elt lst_att '/>' |
               '<'nom_elt lst_att '>' valeur '</' nom_elt '>'
elt_noeud := '<'nom_elt lst_att '>' lst_noeud '</' nom_elt '>'
lst_noeud := arbre | lst_noeud arbre
lst_att := ∅ | att | lst_att att
att := nom_att '= \ ' valeur ' " '
nom_elt := chaîne de caractères
valeur := chaîne de caractères
```

le cnam

Un document XML (suite)

Remarque : au sein d'`elt_feuille` et d'`elt_noeud` toutes les occurrences de `nom_elt` prennent la même valeur.

- des commentaires

3. Éventuellement un épilogue
   - des instructions de traitement
     `<?instruction [attribut = "valeur" ]*? >`
   - des commentaires
     `<!-- texte -->`

The syntax for serialized document, in brief

Four examples of XML documents (separated by blank lines) are :

```
<document/>

<document> Hello World! </document>

<document>
  <salutation> Hello World! </salutation>
</document>

<?xml version="1.0" encoding="utf-8" ?>
<document>
  <salutation color="blue"> Hello World! </salutation>
</document>
```

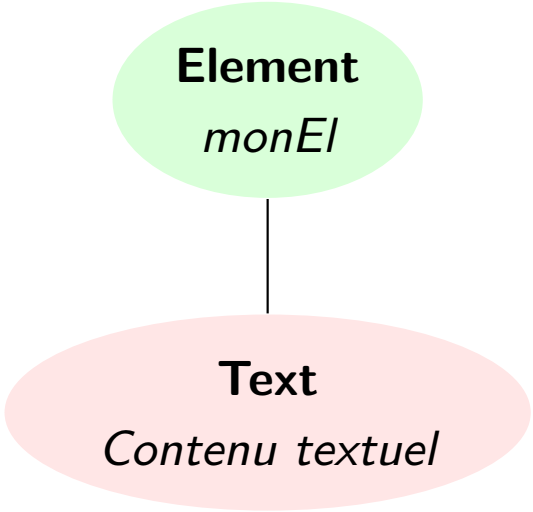
Last example shows the *prologue*, useful for XML parsers (it gives in particular the document encoding).

From serialized to tree form : text and elements

The basic components of an XML document are *element* and *text*.
Here is an *element*, whose content is a *text*.

```
<elt_name>
  Textual content
</elt_name>
```

The tree form of the document, modeled in DOM : each node has a **type**, either **Document** or **Text**.



From serialized to tree form : nesting elements

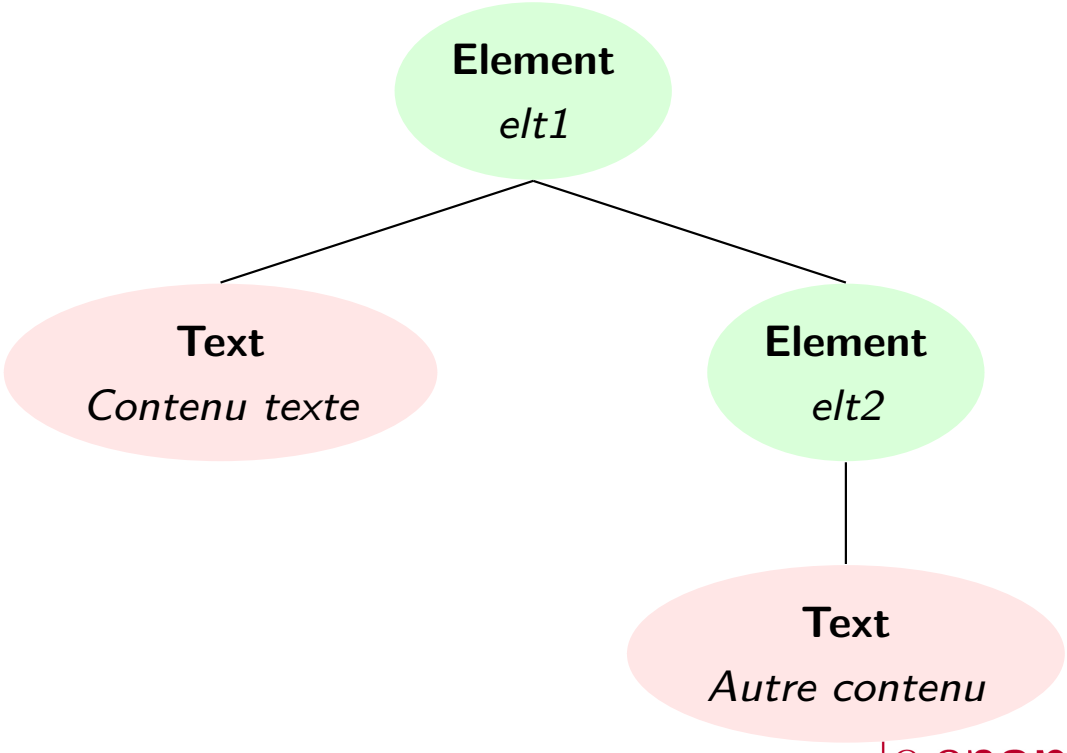
The content of an element is

1. the part between the opening and ending tags (in serialized form),
2. the subtree rooted at the corresponding **Element** node (in DOM).

The content may range from atomic text, to any recursive combination of text and elements (and gadgets, e.g., comments).

Example of an element nested in another element.

```
<elt1>
  Textual  content
  <elt2>
    Another content
  </elt2>
</elt1>
```



le cnam

From serialized to tree form : attributes

Attributes are pairs of name/value attached to an element.

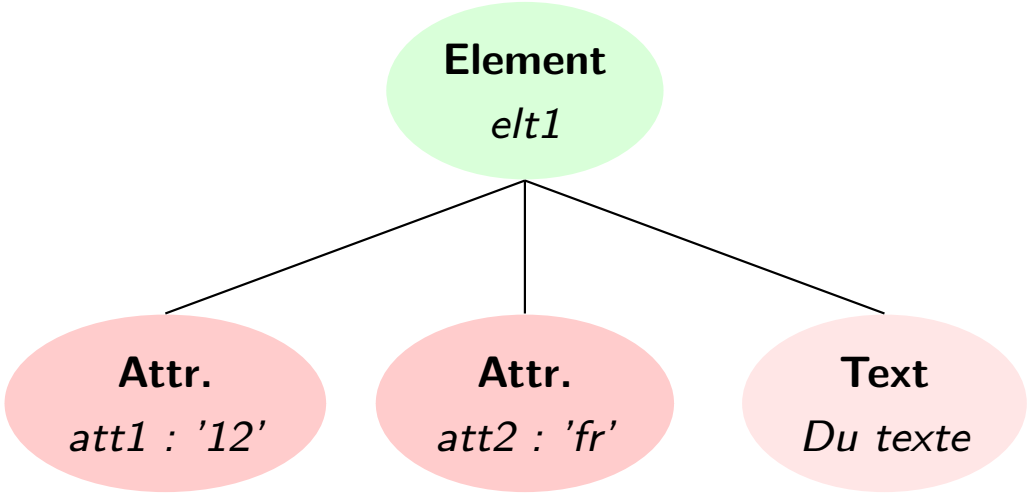
1. as part of the opening tag in the serialized form,
2. as special child nodes of the **Element** node (in DOM).

The content of an attribute is always atomic text (no nesting).

An element with two attributes.

```
<elt1 att1='12' att2='fr'>
  Textual  content
</elt1>
```

Unlike elements, attributes are *not* ordered, and there cannot be two attributes with the same name in an element.



From serialized to tree form : the document root

The first line of the serialized form must *always* be the *prologue* if there is one :

<?**xml version**="1.0" encoding="utf-8" ?>

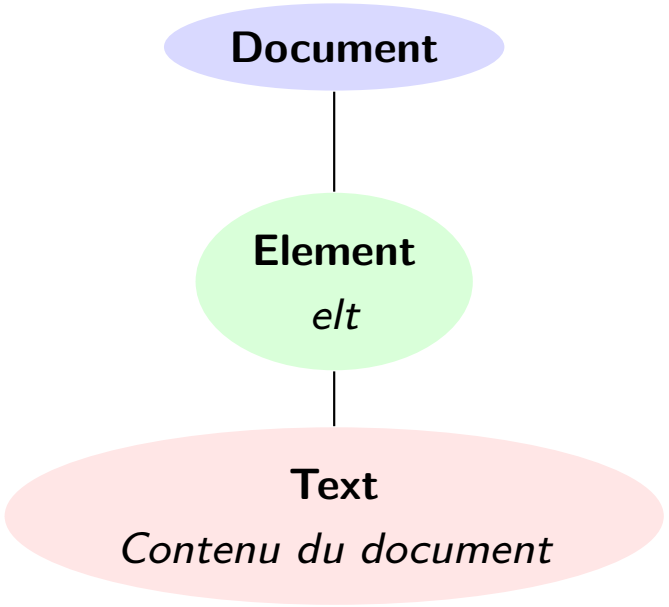
and the document content must *always* be enclosed in a single opening/ending tag, called the *element root*.

A document with its prologue, and element root.

```
<?xml version="1.0"
    encoding="utf-8" ?>
<elt>
  Document content.
</elt>
```

Note : there may be other syntactic objects after the prologue (processing instructions).

In the DOM representation, the prologue appears as a **Document** node, called the *root node*.



le cnam

Summary : syntax and vocabulary

Serialized form

- A document begins with a prologue,
- It consists of a single upper-level tag,
- Each *opening tag* `<name>` has a corresponding *closing tag* `</name>` ; everything between is either text or properly enclosed tag content.

Tree form

- A document is a tree with a *root node* (**Document** node in DOM),
- The root node has one and only one element child (**Element** node in DOM), called the *element root*)
- Each element node is the root of a *subtree* which represents its structured *content*

Summary : syntax and semantics

XML provides a syntax

The core of the syntax is the element name

Element names have no a priori semantics

Applications assign them a semantics

Élément

```
1  <les_genies>
2      <genie>
3          Alan Turing
4      </genie>
5      <genie>
6          Kurt Godel
7      </genie>
8  </les_genies>
```

- `<genie>` est une balise ouvrante (tag ouvrant)
- `</genie>` est une balise fermante (tag fermant)
- `Kurt Godel` est un contenu texte
- `<genie>Alan Turing</genie>` est un élément et un sous-élément de `les_genies`

Élément (suite)

Quelques remarques :

- Les sous-éléments sont ordonnés.
- Le nom du tag reflète généralement le contenu de l'élément.
- Pas de caractères spéciaux (mais -,_,.,/ autorisés) dans les noms des éléments (ni noms d'attributs d'ailleurs), pas d'espaces.
- Un élément peut être vide `<nom_elt></nom_elt>` ou `<nom_elt/>` (Exemples d'éléments vides (XHTML) : `<br/>` ou `<img width="100" height="57" src="logoCnam.jpg"/>`)
- (Contrairement à HTML,) XML est *case sensitive* donc `<genie>` $\neq$ `<Genie>`
- Les caractères spéciaux non autorisés peuvent être remplacés par une référence :
  `&` doit être remplacé par sa référence de caractère `&` (ou par sa référence numérique `&#28;`)
  `<` devient `<` (ou `&#60;`)
  `>` devient `>` (ou `&#62;`)
  `"` devient `&quot;` (ou `&#39;`)
  `'` devient `&apos;` (ou `&#34;`)

Attributs

```
1  <les_genies>
2      <genie date_naiss="1912-06-23" date_deces="1954-06-07">
3          Alan Turing
4      </genie>
5      <genie date_naiss="1906-04-28" date_deces="1978-01-14">
6          Kurt Godel
7      </genie>
8  </les_genies>
```

- `date_naiss` est un attribut du premier élément `<genie>`
- 1912-06-23 est sa valeur

Attributs (suite)

Quelques remarques :

- Zéro ou plusieurs attribut(s) peuvent être associés à un élément.
- Un élément ne peut pas posséder deux attributs de même nom.
- L'ordre des attributs associés à un élément n'a pas d'importance (ils sont identifiés par leur nom).

Document bien formé

Un document est bien formé s'il respecte la syntaxe XML. Un document XML se doit d'être bien formé.

- à tout tag ouvrant est associé un tag fermant,
- les éléments sont imbriqués (pas de superposition),
- présence d'**un seul élément racine**,
- etc (se référer à la syntaxe XML).

Pour tester si l'un de vos documents est bien formé : un vérificateur syntaxique est disponible à l'adresse `http://www.w3schools.com/xml/xml_validator.asp`.
Vous pouvez également ouvrir votre fichier XML avec un navigateur web, intégrant généralement un parser XML (pour pouvoir être parsé, un document doit être bien formé).

Une petite mise en jambes : structuration d'information

Voici quelques informations "en vrac" :

Le britannique Alan Turing est né le 23 juin 1912 à Londres (au Royaume-Uni). Il obtient sa thèse effectuée sous la direction d'Alonzo Church en 1938 à l'université de Princeton, thèse intitulée "Systems of logic defined by ordinals".
Il est célèbre pour ses travaux sur le problème de l'arrêt, les machines de Turing, la Cryptanalyse d'Enigma (pendant la seconde guerre mondiale), le test de turing, la conception et la réalisation de l'ACE (Automatic Computing Engine) réalisé entre 1945 et 1948, au National Physical Laboratory, situé à Teddington au Royaume-Uni. Il reçu d'ailleurs deux distinctions : l'ordre de l'Empire britannique (en 1945, au rang d'officier, officiellement pour ses travaux scientifiques, officieusement pour le décryptage de Enigma), il a est également membre de la Royal Society (élu en 1951, pour ses travaux sur les machines de turing).
Il est à l'origine du prix Turing (informatique), attribué tous les ans depuis 1966.
Il est décédé le 7 juin 1954 (à 41 ans) d'un empoisonnement au cyanure, à Wilmslow (Royaume-Uni).

Kurt Gödel est né le 28 avril 1906 à Brno (Autriche-Hongrie). Il était de nationalité austrau-américaine.
Il obtient son doctorat en philosophie en 1930. Il y prouve la complétude de la logique classique du premier ordre. Il est particulièrement célèbre pour son "théorème d'incomplétude" publié en 1931.
Il est à l'origine du prix Gödel (informatique théorique), attribué tous les ans depuis 1992.
Il est décédé le 14 janvier 1978 de cachexie, à Princeton (États-Unis).

Une petite mise en jambes : structuration d'information (suite)

Test

Réorganisez ces informations afin de les structurer et les présenter sous la forme d'un arbre (ou d'un fichier) XML.

Élément ou attribut ?

Doit-on utiliser un sous-élément ou un attribut pour associer une information à un élément ?

```
<les_genies>
   <genie date_naiss="1912-06-23" date_deces="1954-06-07">
      Alan Turing
   </genie>
</les_genies>
```

ou

```
<genie>
   <nom>Alan Turing</nom>
   <date_naiss>1912-06-23</date_naiss>
   <date_deces>1954-06-07</date_deces>
</genie>
```

Quelle(s) différence(s) d'après vous ?

Plan du cours

Espaces de nom (namespaces)

Les espaces de noms permettent de :

- Lever les ambigüités sur les noms d'éléments et d'attributs
  - Si `nom_elt` de doc_1 et `nom_elt` de doc_2 ne représentent pas la même information
  - Si `nom_elt1` de doc_1 et `nom_elt2` de doc_2 représentent pas la même information mais sont définis différemment.
- Fonction de regroupement des éléments

→ De façon plus générale, associer une **sémantique** aux éléments.

Espaces de nom (namespaces)

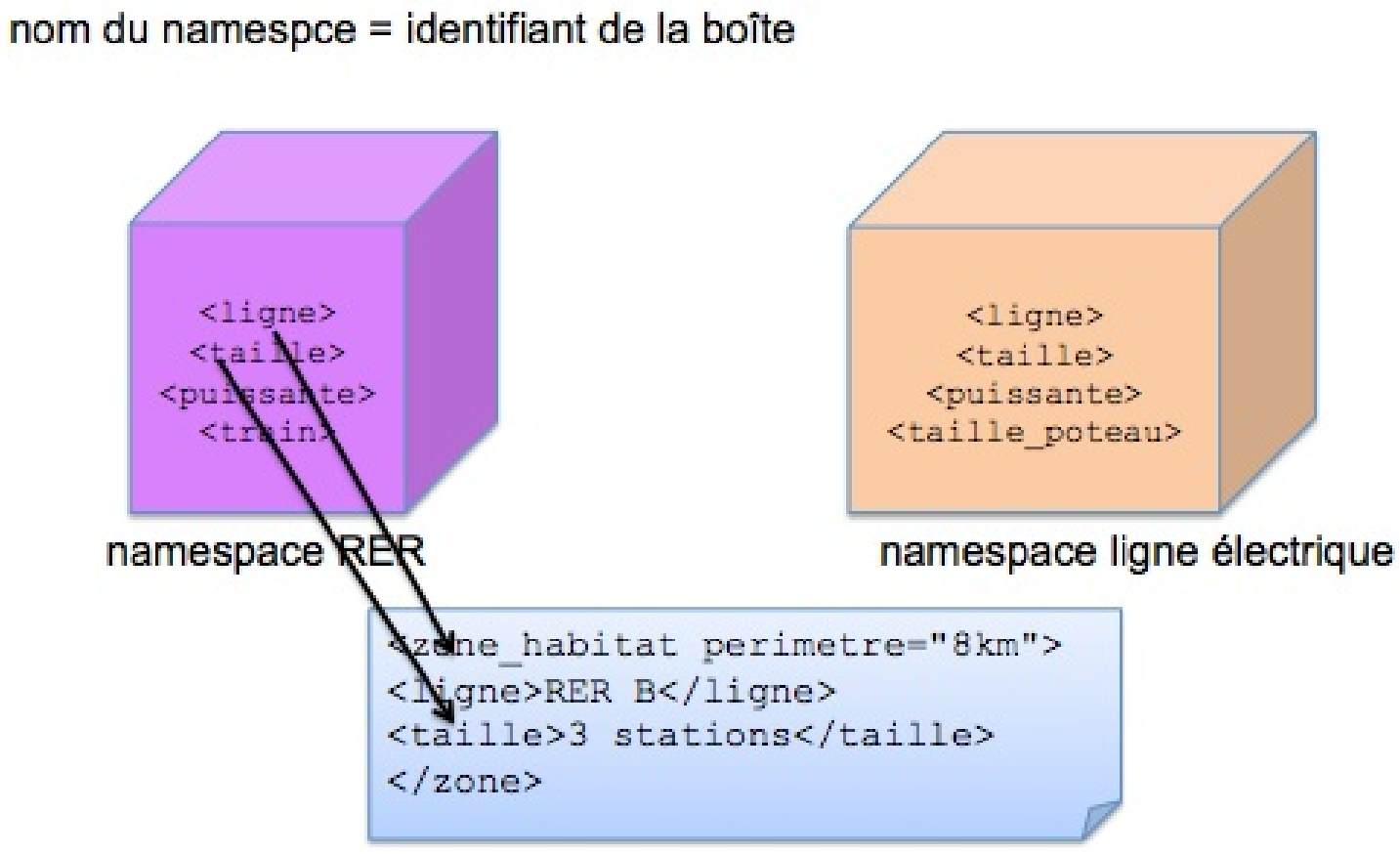


FIGURE 6 – Espace de noms pour associer une sémantique aux éléments

Espaces de nom (namespaces)

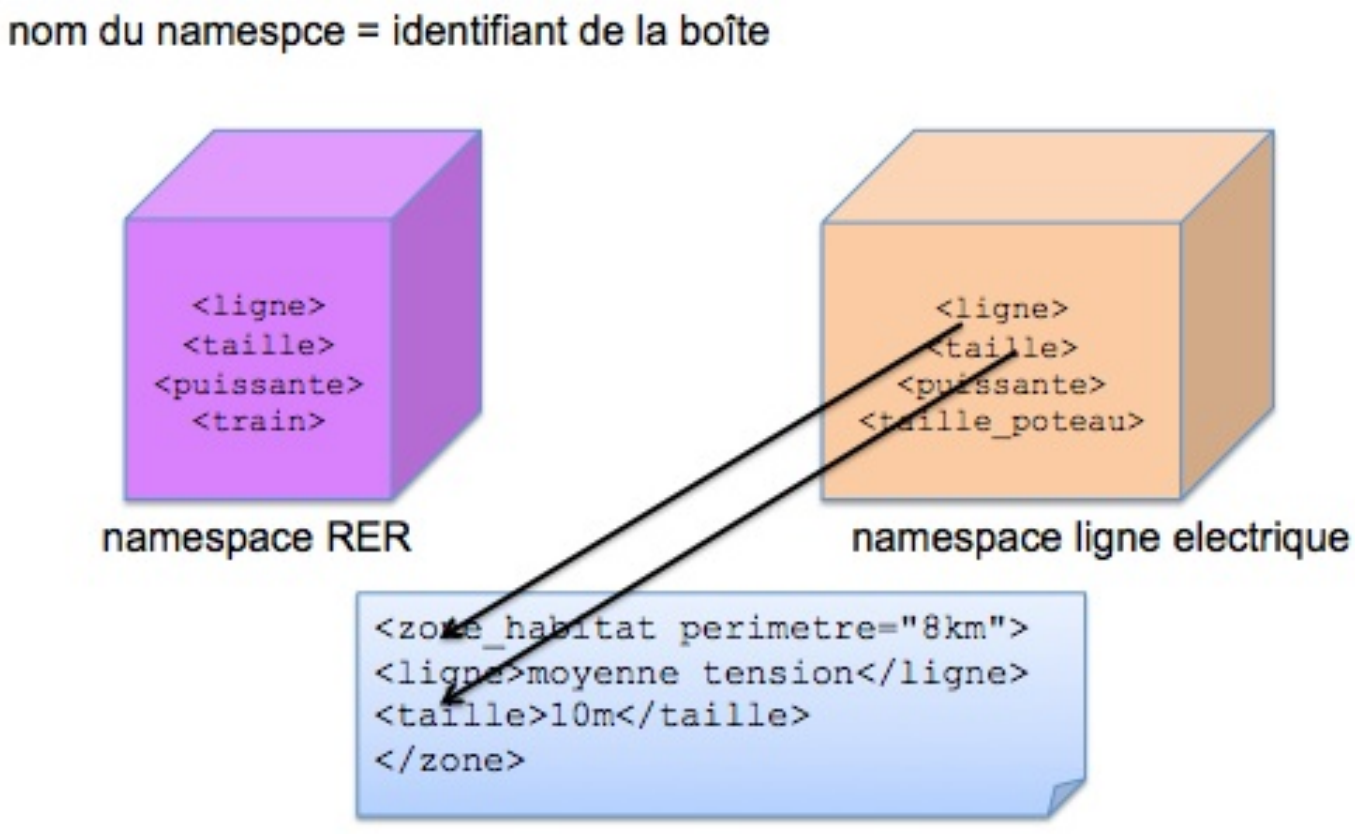


FIGURE 7 – Espace de noms pour associer une sémantique aux éléments

Espaces de nom (namespaces)

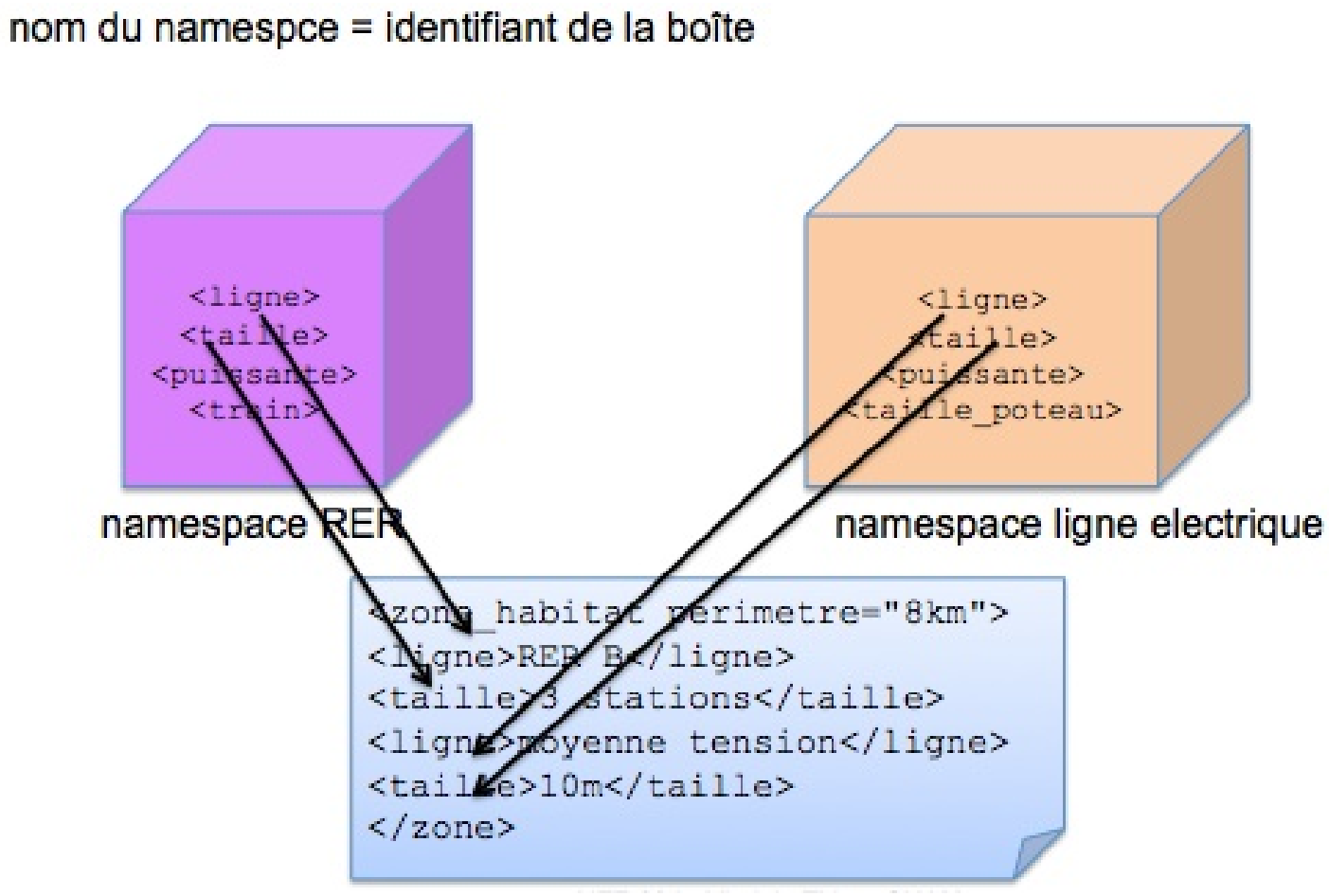


FIGURE 8 – Espace de noms pour associer une sémantique aux éléments

le cnam

Espaces de nom (namespaces)

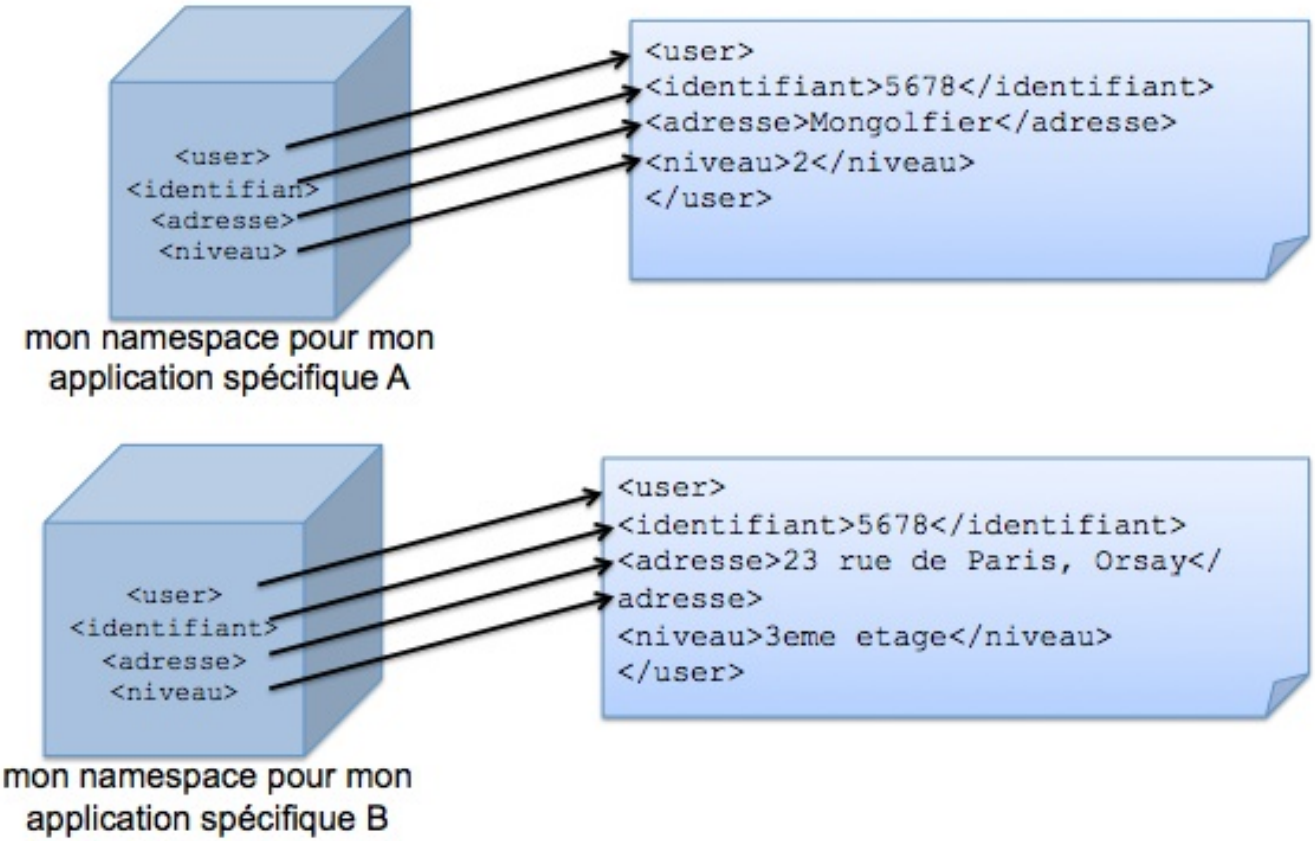


FIGURE 9 – Espace de nom pour associer une information de provenance aux éléments

Déclaration

Il existe deux façons d'associer un espace de noms à un (ou plusieurs éléments) dans un fichier XML :

- par défaut, ou
- explicitement.

Déclaration par défaut

```
<genie xmlns=''http://chezmoi.fr/''>
   <prenom>Alan</prenom>
   <nom>Turing</nom>
</genie>
```

`http://chezmoi.fr/` est le nom (identifiant) de l'espace de noms.

L'**élément genie** et **tous ses fils** sont dans l'espace de nom `http://chezmoi.fr/`. Les textes (Alan et Turing) ne sont dans aucun espace de nom puisque ce sont des *données*. Ne s'applique pas non plus aux noms d'attributs et valeurs d'attributs.

URI (Uniform Resource Identifier) : ce sont généralement des URL.
Pour simplifier, on peut voir cela comme un identifiant "de sémantique" associée aux tags du document

Traduction par un interpréteur

```
<genie xmlns="http://chezmoi.fr/">
   <prenom> Alan </prenom>
   <nom> Turing </nom>
</genie>
```

Ce que comprend (traduit) un interpréteur :

```
<http://chezmoi.fr/genie>
   <http://chezmoi.fr/prenom> Alan </http://chezmoi.fr/prenom>
   <http://chezmoi.fr/nom> Turing </http://chezmoi.fr/nom>
</http://chezmoi.fr/genie>
```

Le *nom étendu* `http://chezmoi.fr/genie` est composé

- du *nom de l'espace de noms* associé à l'élément (`http://chezmoi.fr/`) et
- du *nom local* de l'élément (`genie`) [**?**].

Example de déclaration par défaut

```
<msup xmlns="http://www.w3.org/1998/Math/MathML">
    <mfenced>
        <mrow>
          <mi>a</mi>
          <mo>+</mo>
          <mi>b</mi>
        </mrow>
      </mfenced>
      <mn>2</mn>
    </msup>
```

Listing 6 – Utilisation de l'espace de nom de MathML

Test

Comment ce code est-il traduit par un interpréteur ?

Déclaration explicite

```
<genie xmlns:p=''http://chezmoi.fr/''>
   <prenom>Alan</prenom>
   <nom>Turing</nom>
</genie>
```

Listing 7 – Déclaration explicite d'un espace de nom

Dans le code ci-dessus, l'espace de nom est déclaré mais n'est appliqué à aucun élément ni attribut.
Le préfixe : une fois déclaré dans un élément, il est employable dans l'élément et son contenu. Si un élément ou un attribut n'est pas prefixé, il n'est pas dans l'espace de nom.
Pour l'appliquer :

```
<p:genie xmlns:p=http://chezmoi.fr/ p:date_naiss=''p:1912-06-23''>
   <p:prenom>Alan</p:prenom>
   <nom>Turing</nom>
</p:genie>
```

Listing 8 – Déclaration explicite et application d'un espace de nom

Déclaration explicite

```
<p:genie xmlns:p=http://chezmoi.fr/ p:date_naiss="p:1912-06-23">
   <p:prenom> Alan </p:prenom>
   <nom> Turing </nom>
</p:genie>
```

Ce que traduit l'interpréteur :

```
<http://chezmoi.fr/genie http://chezmoi.fr/date_naiss="http://chezmoi.fr/1912-06-23">
   <http://chezmoi.fr/prenom> Alan </http://chezmoi.fr/prenom>
   <nom> Turing </nom>
</http://chezmoi.fr/genie>
```

Quelques exemples d'utilisation des espaces de noms : les dialectes

```
<math>
 <mrow>
   <msup>
     <mfenced>
       <mrow>
         <mi>a</mi>
         <mo>+</mo>
         <mi>b</mi>
       </mrow>
     </mfenced>
     <mn>2</mn>
   </msup>
 </mrow>
</math>
```

Listing 9 – Sans espace de nom

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
 <mrow>
   <msup>
     <mfenced>
       <mrow>
         <mi>a</mi>
         <mo>+</mo>
         <mi>b</mi>
       </mrow>
     </mfenced>
     <mn>2</mn>
   </msup>
 </mrow>
</math>
```

Listing 10 – Avec espace de nom référençant MathML

Quelques exemples d'utilisation des espaces de noms : les dialectes

Le navigateur Firefox (versions utilisées ici : 4.0.1 et 6.0.1) intègre un interpréteur MathML permettant d'afficher "élégamment" le contenu d'un fichier MathML.

Que se passe-t-il si on ouvre les fichiers contenant les listings 9 et 10 avec Firefox ?

Quelques exemples d'utilisation des espaces de noms : les dialectes

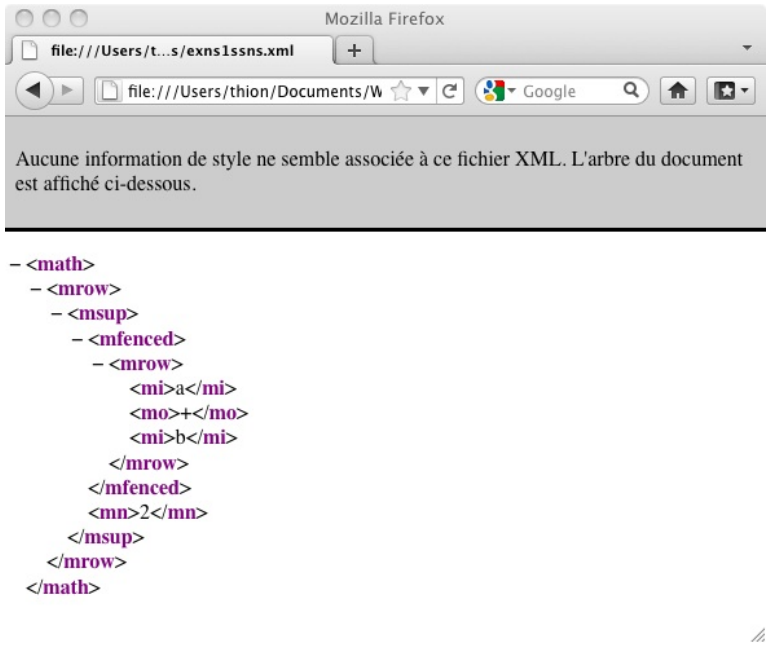


FIGURE 10 – Ouverture du fichier Listing 9 (sans espace de nom) avec Firefox (v. 6.0.1)

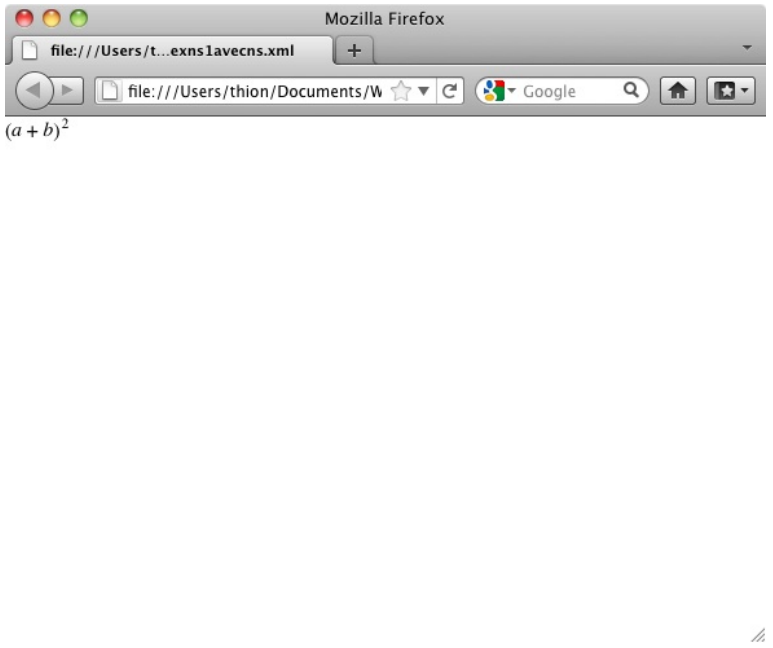


FIGURE 11 – Ouverture du fichier Listing 10 (avec espace de nom) avec Firefox (v. 6.0.1)

le cnam

Quelques exemples d'utilisation des espaces de noms : les dialectes

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <body>
4  Liste courte
5  <ul>
6    <li>Diagramme de classes</li>
7    <li>Diagramme de cas d'utilisation</li>
8    <li>Diagramme de séquence</li>
9    <li>Diagramme d'activité</li>
10   <li>Diagramme d'états-transitions</li>
11   <li>Diagramme de communication</li>
12   <li>Diagramme de composants</li>
13   <li>Diagramme de déploiement</li>
14   <li>Diagramme de structure composite</li>
15   <li>Diagramme global d'interaction</li>
16   <li>Diagramme de packages</li>
17   <li>Diagramme de temps</li>
18 </ul>
```

Listing 11 – Extrait (premières lignes) d'un fichier xml

Existe-t-il un espace de noms déclaré ? Si oui, lequel et à quel(s) élément(s) est-il appliqué ?

Que se passe-t-il si on ouvre ce fichier avec Firefox ?

Quelques exemples d'utilisation des espaces de noms : les dialectes

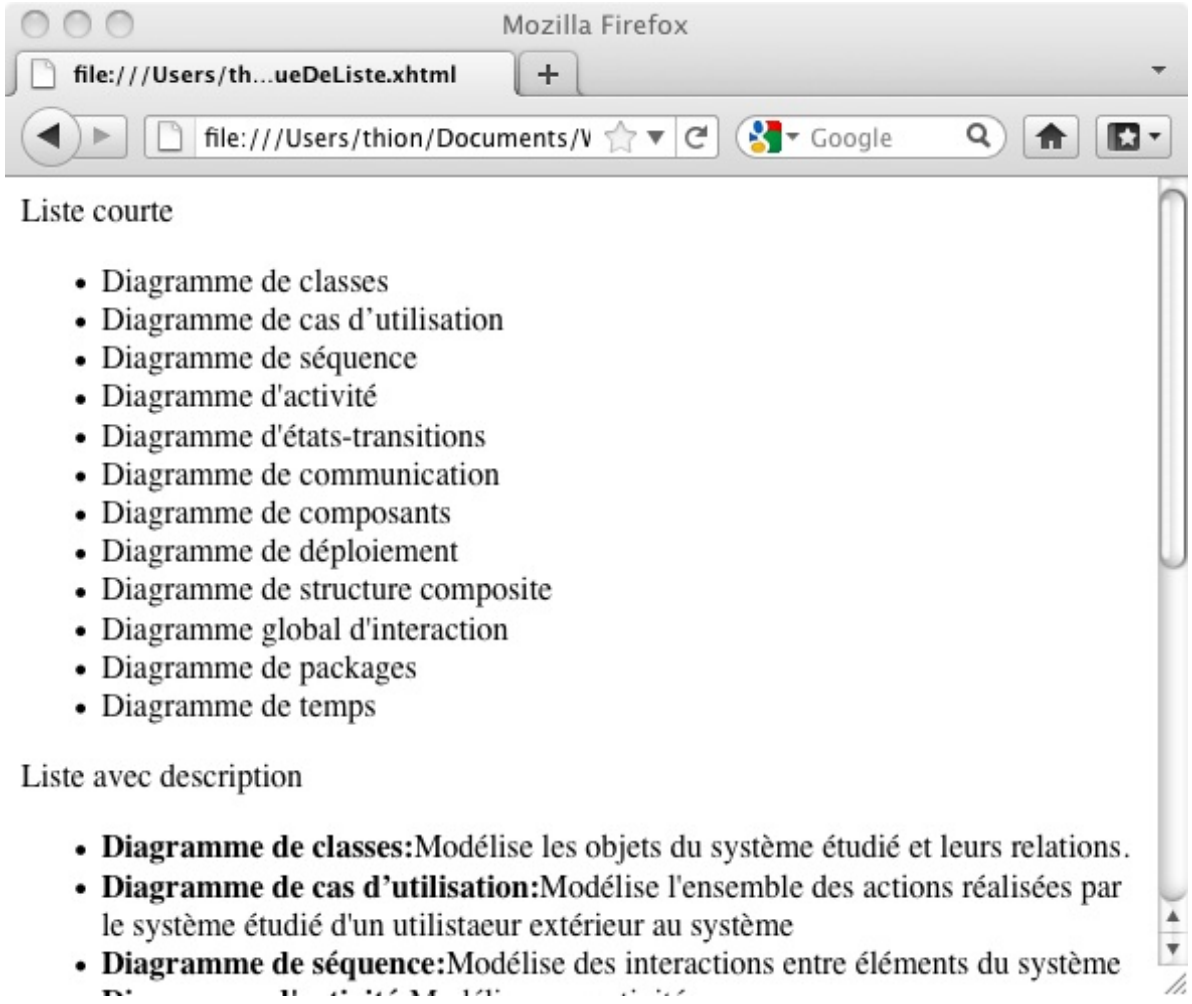


FIGURE 12 – Fichier contenant le Listing 11, ouvert sous Firefox (v 6.0.1)

le cnam

Quelques exemples d'utilisation des espaces de noms : les dialectes

---

Que se passe-t-il lors de l'ouverture sous Firefox si la ligne 2 du listing 11 est remplacée par la ligne suivante ?


```
2   <html>
```

Les URI

Les URL

Les URL utilisées comme URI ne pointent pas forcément vers une page existante. Elles servent juste de "chaîne de caractères identifiant".

Pourtant,

- La doc de XHTML se trouve sur `http://www.w3.org/TR/xhtml1/`. Le namespace (identifiant) de XML est `http://www.w3.org/1999/xhtml/`.
- si vous rendez sur `http://www.w3.org/1999/xhtml/`, vous trouvez une page Web contenant des informations concernant le dialecte XHTML :

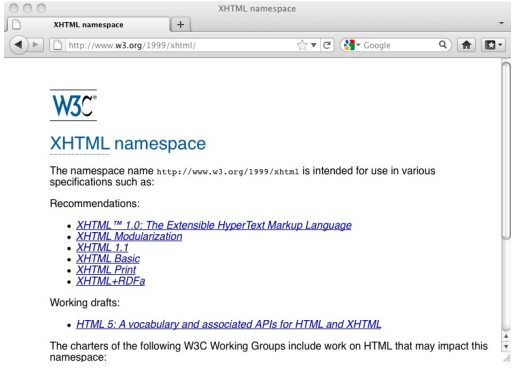


FIGURE 13 – Page `http://www.w3.org/1999/xhtml/`

A votre avis, pourquoi ?

Plan du cours

La validation

Valider :

1. Associer un schéma permettant d'ajouter des contraintes sur les données :
   - noms des éléments,
   - sous-éléments possibles d'un élément, ordre d'apparition et nombre d'occurrences,
   - attributs d'un élément,
   - type : chaîne, type énuméré, clef, clef étrangère,
   - occurence : obligatoire ou non,
   - valeur : par défaut ou fixée,
   - etc.

   Pour les documents XML, les schémas sont exprimés sous la forme d'une grammaire.
2. Vérifier que le document est conforme à la grammaire.

Objectifs

Assurer une meilleure qualité des données transmises ou reçues [a].
Améliorer l'interopérabilité des applications.

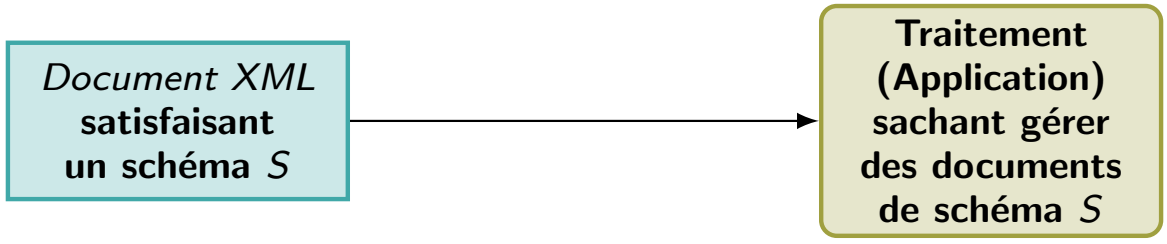
a. Évidemment, associer un schéma n'est généralement pas suffisant pour assurer une bonne qualité des données mais le schéma y participe

La validation (suite)

Document valide

Un document est *valide* s'il respecte le schéma qui lui est associé.

Valider un document XML



Valider un document XML (suite)

Valider un document XML

Formalismes permettant d'écrire un schéma de document(s) XML :

- avec une syntaxe particulière
  - DTD (Document Type Definition)
- dans un dialecte XML
  - XML Schema
  - RelaxNG,
  - Schematron,
  - ...

Schéma exprimé sous forme d'une grammaire, décrivant la "forme" des arbres XML satisfaisant le schéma.

Valider un document XML

Formalismes permettant d'écrire un schéma de document(s) XML :

- avec une syntaxe particulière
  - DTD (Document Type Definition) ← *Dans la norme XML elle-même (donc recommandé W3C)*
- dans un dialecte XML
  - XML Schema
  - RelaxNG,
  - Schematron,
  - ...

Schéma exprimé sous forme d'une grammaire, décrivant la "forme" des arbres XML satisfaisant le schéma.

Valider un document XML

Formalismes permettant d'écrire un schéma de document(s) XML :

- avec une syntaxe particulière
  - DTD (Document Type Definition) ← *Dans la norme XML elle-même (donc recommandé W3C)*
- dans un dialecte XML
  - XML Schema ← *Recommandé W3C*
  - RelaxNG,
  - Schematron,
  - ...

Schéma exprimé sous forme d'une grammaire, décrivant la "forme" des arbres XML satisfaisant le schéma.

Valider un document XML

Formalismes permettant d'écrire un schéma de document(s) XML :

- avec une syntaxe particulière
  - **DTD (Document Type Definition)** ← *Dans la norme XML elle-même (donc recommandé W3C)*
- dans un dialecte XML
  - **XML Schema** ← *Recommandé W3C*
  - RelaxNG,
  - Schematron,
  - ...

Schéma exprimé sous forme d'une grammaire, décrivant la "forme" des arbres XML satisfaisant le schéma.

Une DTD en deux mots

Sous forme d'une grammaire :

- ⇗ simple à écrire et à comprendre,
- ⇗ rapide à écrire,
- ⇘ peu expressive.

```
1  <!ELEMENT NEWSPAPER (ARTICLE+)>
2  <!ELEMENT ARTICLE (AUTHOR+,HEADLINE, BYLINE, LEAD, BODY, NOTES*)>
3  <!ELEMENT HEADLINE (#PCDATA)>
4  <!ELEMENT AUTHOR(#PCDATA)>
5  <!ELEMENT BYLINE (#PCDATA)>
6  <!ELEMENT LEAD (#PCDATA)>
7  <!ELEMENT BODY (#PCDATA)>
8  <!ELEMENT NOTES (#PCDATA)>
9
10 <!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
11 <!ATTLIST ARTICLE DATE CDATA #IMPLIED>
12 <!ATTLIST ARTICLE EDITION CDATA #IMPLIED>
```

Listing 12 – Exemple simple d'une DTD

Définition d'un élément

- élément vide `<!ELEMENT nom_elt EMPTY>`

Élément vide

Bout de la DTD : `<!ELEMENT br EMPTY>`
Un élément le satisfaisant : `<br/>` ou `<br></br>`

Test

L'élément `<BR/>` satisfait-il le bout de DTD ci-dessus ?

- feuille `<!ELEMENT nom (#PCDATA)>`

Feuille

Bout de la DTD : `<!ELEMENT auteur (#PCDATA)>`
Un élément le satisfaisant : `<auteur>Victor Hugo</auteur>`

Test

Proposez un autre élément satisfaisant le bout de DTD ci-dessus.

cnam

Définition d'un élément (suite)

- sans contrainte sur les sous-éléments `<!ELEMENT nom ANY>`

Sans contrainte sur les sous-éléments

Bout de la DTD : `<!ELEMENT auteur ANY)>`
Un élément le satisfaisant : `<auteur>Victor Hugo</auteur>`
Un autre élément le satisfaisant :
`<auteur><prenom>Victor</prenom><nom>Hugo</nom></auteur>`

Test

Proposez un autre élément satisfaisant le bout de DTD ci-dessus.

Définition d'un élément (suite)

- avec contrainte sur les sous-éléments

Avec contrainte sur les sous-éléments

Bouts de DTD : `<!ELEMENT auteur (nom)>`

```
<!ELEMENT auteur (nom?)>
<!ELEMENT auteur (livre*)>
<!ELEMENT auteur (nom, prenom)>     ← séquence
<!ELEMENT cours (magistral | projet)>     ← choix
<!ELEMENT autheur (nom, prenom?)>
<!ELEMENT livre (isbn,sommaire?)>
<!ELEMENT livre (isbn,titre,auteur+,critiques*)>
<!ELEMENT cours ((date,salle)+,(url|fichier+)+)>
```

`<!ELEMENT nom children>` où
`children := (seq | choice)` cardinalité
`choice := '(' cp ('|' cp)+ ')'`
`seq := '(' cp (',' cp)* ')'`
`cp := (nom | choice | seq)` cardinalité
cardinalités :

(vide)	: exactement 1 occurrence
*	: 0 à ∞ occurrences
+	: 1 à ∞ occurrences
?	: 0 ou 1 occurrence

Contraintes sur la DTD :
Il doit exister au plus une déclaration `<!ELEMENT ..>` par élément.
Tous les éléments doivent être définis.

Définition d'un élément (suite)

Test

1. Proposez des éléments satisfaisant chacun des bouts de DTD ci-dessus.
2. Proposez quelques définitions d'éléments avec contraintes sur les sous-éléments, et quelques éléménts intance qui pourraient satisfaire chacun d'eux.
3. Proposez deux définitions d'élément que satisfait un même élément instance.

Pour avoir la définition complète d'une DTD : [?]

Simples DTD

Test

Ajoutez les cardinalités à la DTD suivante :

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Simples DTD

Test

Ajoutez l'information *une note contient soit un heading soit un body* à la DTD ci-dessous :

```
<!ELEMENT note (to+, from, heading?, body?)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Listing 13 – Le memo

Définition d'attributs

Rappel :

```
<genie date_naiss="1906-04-28" date_deces="1978-01-14">
    Kurt Godel
</genie>
```

Listing 14 – Élement `genie` et ses attributs

Pour déclarer un attribut :

```
<!ATTLIST nom_elt
 nom_att_1 type_1 occurrence_1 default_1
 ...
 nom_att_n type_n occurrence_n default_n>
```

```
<!ATTLIST book
    ISBN CDATA #REQUIRED
    lang (fr|en) #IMPLIED
    subject CDATA #IMPLIED>
```

Définition d'attributs (suite)

```
<!ATTLIST nom_elt
 nom_att_1 type_1 occurrence_1 default_1
 ...
 nom_att_n type_n occurrence_n default_n>
```

<!**ATTLIST** book
ISBN **CDATA** #**REQUIRED**
lang (fr|en) #**IMPLIED**
subject **CDATA** #**IMPLIED**>

où

- `nom_elt` est le nom de l'élément possédant les attributs
- `nom_att_i` $(1 \leq \mathtt{i} \leq \mathtt{n})$ est le nom d'un attribut défini.
  Contrainte : $\nexists \{i,j\} \subseteq \{1,\cdots,n\}$ *with* $i \neq j \mid nom\_att\_i = nom\_att\_j$
- `type_i` $(1 \leq \mathtt{i} \leq \mathtt{n})$ est le type de l'attribut *nom_att_i*, parmis
  - `CDATA` : chaîne de caractères (une fois instancié, il s'agit de la valeur de l'attribut)
  - $(\mathtt{ch}_1|...|\mathtt{ch}_\mathtt{k})$: énumération des chaînes de caractères
  - `ID` : clef
  - `IDREF` : clef étrangère référençant une clef
  - `IDREFS` : clef étrangère référençant des clef
  - ...

  Contrainte : $\nexists \{i,j\} \subseteq \{1,\cdots,n\}$ *with* $i \neq j \mid type\_i = ID$ *and* $type\_i = type\_j$

le cnam

Définition d'attributs (suite)

- `occurence_i` ($1 \leq i \leq n$) est l'occurence associée à l'attribut `nom_att_i` et prend sa valeur dans :

  `#REQUIRED` = obligatoire
  `#IMPLIED` = optionnel
  `#FIXED valeur` = valeur fixée (pas de valeur par défaut)
  (vide) : par défaut, `#REQUIRED`

  Contrainte pour la DTD : si `type_i` ($1 \leq i \leq n$) = *ID* alors `occurrence_i` $\in \{$`#REQUIRED`, `#IMPLIED`$\}$.

- `default` est la valeur par défaut (si `#FIXED`, `#IMPLIED` ou énumération)
  default = valeur par défaut

Déclaration d'attributs

Pour une DTD,

1. Il ne doit exister qu'une déclaration d'élément `<!ELEMENT ..>` par élément.
2. Il peut exister plusieurs définitions `ATTLIST` pour un même élément mais elles sont considérées comme une seule (fusion des définitions en une seule).

DTD complète avec attribut

```
1  <!DOCTYPE biblio [
2          <!ELEMENT biblio (book*)>
3          <!ATTLIST biblio
4                  subject CDATA #IMPLIED>
5          <!ELEMENT book (author+,title,publisher,pubyear)>
6          <!ATTLIST book
7                  ISBN CDATA #REQUIRED
8                  lang (fr|en) #IMPLIED
9                  subject CDATA #IMPLIED
10                 abstract CDATA #IMPLIED>
11         <!ELEMENT author (firstname,lastname)>
12         <!ELEMENT publisher (name,place)>
13         <!ELEMENT name (#PCDATA)>
14         <!ELEMENT place (#PCDATA)>
15         <!ELEMENT title (#PCDATA)>
16         <!ELEMENT firstname (#PCDATA)>
17         <!ELEMENT lastname (#PCDATA)>
18         <!ELEMENT pubyear (#PCDATA)>
19 ]>
```

Listing 15 – DTD bibliothèque

Test

Proposez une instance de cette DTD.

Souvenez-vous : Alan et Kurt

Le britannique Alan Turing est né le 23 juin 1912 à Londres (au Royaume-Uni). Il obtient sa thèse effectuée sous la direction d'Alonzo Church en 1938 à l'université de Princeton, thèse intitulée "Systems of logic defined by ordinals".
Il est célèbre pour ses travaux sur le problème de l'arrêt, les machines de Turing, la Cryptanalyse d'Enigma (pendant la seconde guerre mondiale), le test de turing, la conception et la réalisation de l'ACE (Automatic Computing Engine) réalisé entre 1945 et 1948, au National Physical Laboratory, situé à Teddington au Royaume-Uni. Il reçu d'ailleurs deux distinctions : l'ordre de l'Empire britannique (en 1945, au rang d'officier, officiellement pour ses travaux scientifiques, officieusement pour le décryptage de Enigma), il a est également membre de la Royal Society (élu en 1951, pour ses travaux sur les machines de turing).
Il est à l'origine du prix Turing (informatique), attribué tous les ans depuis 1966.
Il est décédé le 7 juin 1954 (à 41 ans) d'un empoisonnement au cyanure, à Wilmslow (Royaume-Uni).

Kurt Gödel est né le 28 avril 1906 à Brno (Autriche-Hongrie). Il était de nationalité austrau-américaine.
Il obtient son doctorat en philosophie en 1930. Il y prouve la complétude de la logique classique du premier ordre. Il est particulièrement célèbre pour son "théorème d'incomplétude" publié en 1931.
Il est à l'origine du prix Gödel (informatique théorique), attribué tous les ans depuis 1992.
Il est décédé le 14 janvier 1978 de cachexie, à Princeton (États-Unis).

Test

Quelle DTD de texte vous inspirerait-il ? En d'autres termes, quelle DTD mettriez-vous en œuvre si vous deviez créer un "système" stockant des informations concernant les grands génies de la science ?
Vous pouvez vous aider de l'instance que vous avez élaborée précédemment.

ID et IDREF

Rappel (cycles précédents).

Clef Une clef est un attribut qui distingue des "éléments" entre eux. Des exemples classiques de clef sont le numéro de sécurité sociale, le numéros ISBN d'un libre, le numéro d'une carte bleue, ou encore la référence des produits dans un catalogue de vente par correspondance.
En XML, une clef est de type `ID`*. La valeur de cet ID doit être unique (parmis tous les ID)* **dans le document**.

Clef étrangère Une clef étrangère est un attribut prenant la valeur d'une clef. Un exemple classique : le commande client comporte des lignes de commandes pour lesquelles chaque ligne de commande est composée d'une clef étrangère vers une référence produit du catalogue et d'une quantité commandée de ce produit.
En XML, une clef étrangère (resp. un ensemble de clefs étrangères) est (resp. sont) déclarée(s) dans un attribut de type IDREF (resp. IDREFS).

Attention, les notions de clef au sens XML et de clef au sens des bases de données diffèrent (questions de typage/portées de la clef) !

le cnam

DTD complète avec attribut

```
1  <!DOCTYPE biblio [
2          <!ELEMENT biblio (book*)>
3          <!ATTLIST biblio
4                  subject CDATA #IMPLIED>
5          <!ELEMENT book (author+,title,publisher,pubyear)>
6          <!ATTLIST book
7                  ISBN CDATA #REQUIRED
8                  lang (fr|en) #IMPLIED
9                  subject CDATA #IMPLIED
10                 abstract CDATA #IMPLIED>
11         <!ELEMENT author (firstname,lastname)>
12         <!ELEMENT publisher (name,place)>
13         <!ELEMENT name (#PCDATA)>
14         <!ELEMENT place (#PCDATA)>
15         <!ELEMENT title (#PCDATA)>
16         <!ELEMENT firstname (#PCDATA)>
17         <!ELEMENT lastname (#PCDATA)>
18         <!ELEMENT pubyear (#PCDATA)>
19 ]>
```

Listing 16 – DTD bibliothèque

Test

Auquel (Auxquels) des attributs ci-dessous vous semble-t-il pouvoir associer le type ID ?

DTD complète avec attribut

```
1  <!DOCTYPE biblio[
2          <!ELEMENT biblio (book*)>
3          <!ATTLIST biblio
4                  subject CDATA #IMPLIED>
5          <!ELEMENT book (author+,title,publisher,pubyear)>
6          <!ATTLIST book
7                  ISBN ID #REQUIRED
8                  lang (fr|en) #IMPLIED
9                  subject CDATA #IMPLIED
10                 abstract CDATA #IMPLIED>
11         <!ELEMENT author (firstname,lastname)>
12         <!ELEMENT publisher (name,place)>
13         <!ELEMENT name (#PCDATA)>
14         <!ELEMENT place (#PCDATA)>
15         <!ELEMENT title (#PCDATA)>
16         <!ELEMENT firstname (#PCDATA)>
17         <!ELEMENT lastname (#PCDATA)>
18         <!ELEMENT pubyear (#PCDATA)>
19 ]>
```

Listing 17 – DTD bibliothèque avec ID

Portée d'un identifiant en XML

Spécifie un ID unique au sein du document XML entier, dont la portée est tout le document → Il ne doit pas exister deux valeurs identiques d'ID (tous élements confondus) au sein d'un même document (nous ne sommes pas habitués à cela dans le monde des BD).

IDREF(S)

Revenons sur la définition de IDREF :

```
<!ATTLIST nom_elt
 nom_att_1 type_1 occurrence_1 default_1
 ...
 nom_att_n type_n occurrence_n default_n>
```

Par exemple, (bout de DTD) :

```
1  <!ELEMENT UE (nom, nb_credits?)>
2  <!ATTLIST UE
3     num_UE ID #REQUIRED,
4     enseignant IDREF #REQUIRED>
5  <!ELEMENT nom (#PCDATA)>
6  <!ELEMENT nb_credits (#PCDATA)>
```

Listing 18 – DTD bibliothèque avec ID

Un exemple de clef étrangère déclarée en SQL :

```
1  CREATE TABLE UE
2  (
3   id INTEGER NOT NULL,
4   nom VARCHAR2(15) NOT NULL,
5   nb_credits INTEGER,
6   id_ens INTEGER,
7   CONSTRAINT UE_pkey PRIMARY KEY (id),
8   CONSTRAINT UE_id_responsable_fkey FOREIGN KEY (id_ens) REFERENCES ENSEIGNANT (id)
9  ) ;
```

Listing 19 – Clef étrangère déclarée en SQL

IDREF(S)

Ma DTD contient (entre autres) les déclarations d'attributs suivantes (en admettant que les éléments soient déclarés avant) :

```
1  <!ATTLIST square width CDATA ''0''>
2  ..
3  <!ATTLIST voiture carburant (essence|diesel|hybride|electrique)>
4  ..
5  <!ATTLIST person number CDATA #REQUIRED>
6  ..
7  <!ATTLIST contact fax CDATA #IMPLIED>
8  ..
9  <!ATTLIST sender company CDATA #FIXED ''Microsoft''>
10 ..
11 <!ATTLIST employe numsecu ID #REQUIRED>
12 ..
13 <!ATTLIST livre isbn ID #REQUIRED>
14 ..
15 <!ATTLIST assure person IDREF #REQUIRED>
16 ..
```

Puis-je trouver les lignes suivantes dans un fichier XML vérifiant la DTD ?

```
17 <square width=''100" />
18 ..
19 <person number="5677" />
20 ..
21 <person />
22 ..
23 <contact fax="555-667788" />
24 ..
25 <contact />
26 ..
27 <sender company="Microsoft" />
28 ..
29 <sender company="W3Schools" />
30 ..
31 <employee numsecu="1760413"/>
32 ..
33 <livre isbn="978-0070507753"/>
34 ..
35 <assure person="1760413">
36 ..
37 <employee numsecu="978-0070507753">
38 ..
```

le cnam

IDREF(S) (suite)

```
1  ..
2  <!ELEMENT livre>
3  <!ATTLIST livre isbn ID #REQUIRED>
4  <!ELEMENT employe>
5  <!ATTLIST employe numsecu ID #REQUIRED>
6  <!ELEMENT vendeur>
7  <!ATTLIST vendeur person IDREF #REQUIRED>
```

Listing 20 – Bout de déclaration dans une DTD

```
1  ..
2  <employe numsecu=''1760413...''/>
3  <livre isbn=''978-0070507753''/>
4  <vendeur person=''1760413...''>
5  <vendeur person=''978-0070507753''>
6  ..
```

Listing 21 – Bout d'instance

Test

Le bout de document du listing 20 vous semble-t-il valide par rapport au bout de DTD du listing 21 ? Que remarquez-vous ?

IDREF(S) (suite)

```
1  ..
2  <!ATTLIST livre isbn ID #REQUIRED>
3  ..
```

Listing 22 – Bout de déclaration dans une DTD

```
1  ..
2  <livre isbn="978-0070507753" />
3      <titre>Au bonheur des dames</titre>
4      <auteur>Emile Zola</auteur>
5  </livre>
6  <livre isbn="978-0070507753" />
7      <titre>L'Ecume des jours</titre>
8      <auteur>Boris Vian</auteur>
9  </livre>
10 ..
```

Listing 23 – Bout d'instance

Test

1. Puis-je trouver dans un fichier XML vérifiant la DTD du listing 22, les éléments du listing 23 ?
2. Puis-je trouver ces éléments dans un fichier bien formé ?

Remue-méninges au cinéma...

Test

1. Choisissez un film (f1), ainsi que deux de ses acteurs (a1 et a2).
2. Choisissez maintenant un autre film (f2) dans lequel joue a1.
3. Un film est composé d'un titre, d'une année, d'un réalisateur, et d'une liste d'acteurs. Un acteur est composé d'un nom, d'une date de naissance, et de la liste des films dans lesquels il joue.

Proposez ensuite un fichier XML et sa DTD associée permettant de représenter ces informations avec le moins de redondance possible.

En manque d'inspiration ?

Harrison Ford et Larry Fishburne ont tous les deux joué dans Apocalypse Now (1979, réalisation de Francis Ford Coppola).
Harrison Ford a également joué dans Star Wars (1977, réalisation de George Lucas).

FIGURE 14 – Larry Fishburne (né le 30/07/1965)

FIGURE 15 – Harrison Ford (né le 13/07/1942)

Remue-méninges au cinéma...

Association plusieurs-à-plusieurs entre les entités film et acteur :

- à un acteur sont associés plusieurs films
- à un film sont associés plusieurs acteurs

Le problème : fichier *films_le_pb.xml* (fichier disponible sur Plei@d)
Une correction : fichier *films_correction.xml* (fichier disponible sur Plei@d)

Déclarer une DTD et l'associer à un (ou plusieurs) document(s)

Deux façons de déclarer :

- Déclaration interne (au fichier de données), ou
- Déclaration externe (au fichier de données)

Déclarer une DTD et l'associer à un (ou plusieurs) document(s) (suite)

Déclaration **interne** : la DTD est définie dans le document lui-même.

`<!DOCTYPE root-element [element-declarations]>`

```
1  <?xml version="1.0"?>
2
3  <!DOCTYPE note [
4            <!ELEMENT note (to,from,heading,body)>
5            <!ELEMENT to (#PCDATA)>
6            <!ELEMENT from (#PCDATA)>
7            <!ELEMENT heading (#PCDATA)>
8            <!ELEMENT body (#PCDATA)>
9   ]>
10
11 <note>
12   <to>Tove</to>
13   <from>Jani</from>
14   <heading>Reminder</heading>
15   <body>Don't forget me this weekend</body>
16 </note>
```

Listing 24 – Déclaration interne de DTD

Déclarer une DTD et l'associer à un (ou plusieurs) document(s)

Déclaration **externe** : la DTD est définie dans un autre document, appelé dans le document contenant les données.

`<!DOCTYPE root-element SYSTEM "filename">`

```
1  <!ELEMENT note (to,from,heading,body)>
2  <!ELEMENT to (#PCDATA)>
3  <!ELEMENT from (#PCDATA)>
4  <!ELEMENT heading (#PCDATA)>
5  <!ELEMENT body (#PCDATA)>
```

Listing 25 – DTD contenue dans le fichier note.dtd

```
1  <?xml version="1.0"?>
2  <!DOCTYPE note SYSTEM "note.dtd">
3  <note>
4    <to>Tove</to>
5    <from>Jani</from>
6    <heading>Reminder</heading>
7    <body>Don't forget me this weekend!</body>
8  </note>
```

Listing 26 – Déclaration externe, fichier appelant la DTD ci-dessus

La ligne 2 indique au processeur 1) qu'une DTD est associée et 2) où la trouver.

Un exemple d'utilisation de DTD : MathML

```
1  <?xml version="1.0"?>
2  <math>
3   <mrow>
4     <msup>
5       <mfenced>
6         <mrow>
7           <mi>a</mi>
8           <mo>+</mo>
9           <mi>b</mi>
10        </mrow>
11      </mfenced>
12      <mn>2</mn>
13    </msup>
14  </mrow>
15 </math>
```

Un exemple d'utilisation de DTD : MathML (suite)

Vérifier qu'un document est conforme à la syntaxe MathML ?

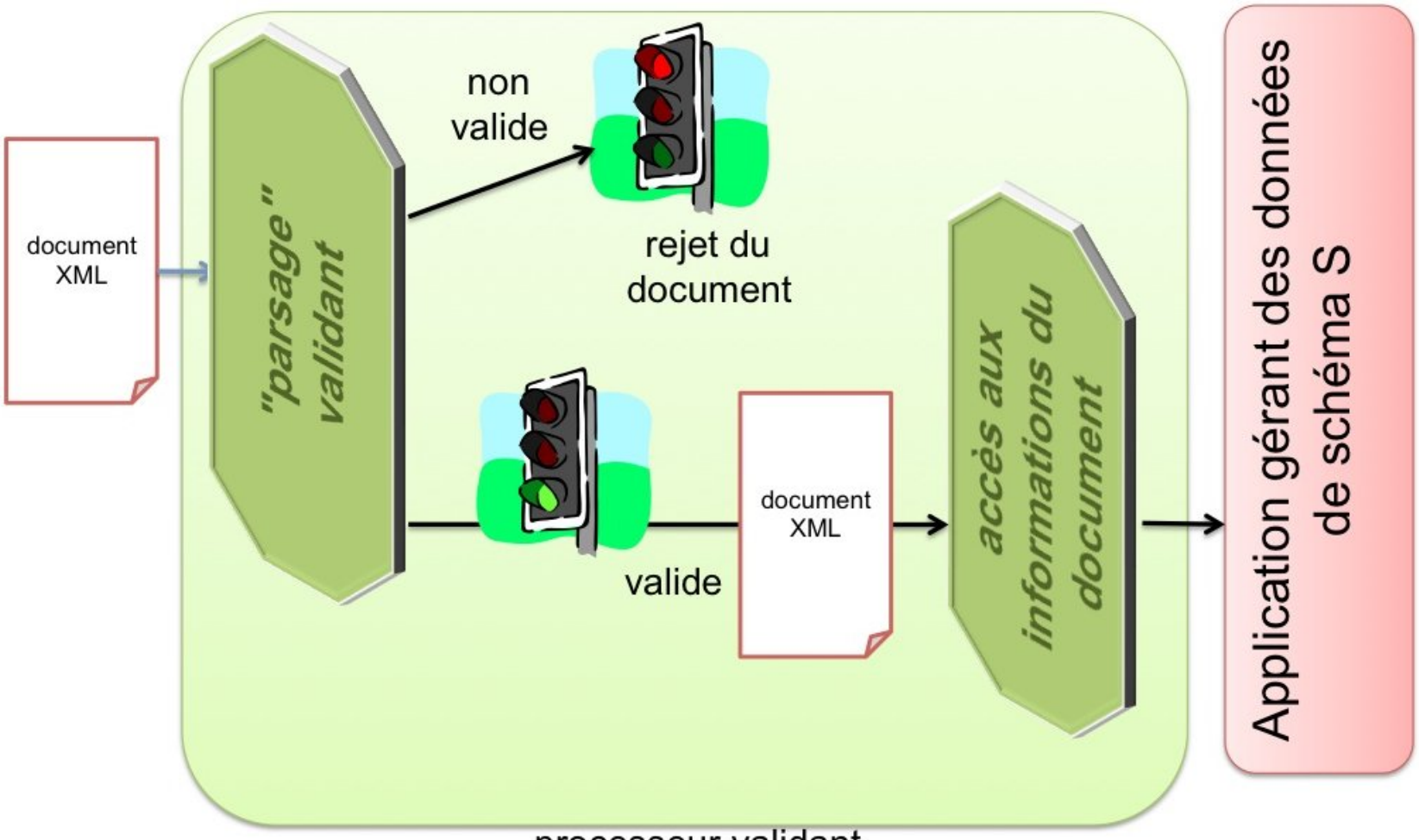
- La DTD de MathML a été définie et mise en ligne à `http://www.w3.org/TR/MathML2/dtd/mathml2.dtd`
- Pour s'assurer qu'un document XML est conforme à cette DTD, on peut par exemple déclarer la DTD dans de MathML celui-ci
  `<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"`
  `http://www.w3.org/TR/MathML2/dtd/mathml2.dtd>`
- On teste la validité du document à l'aide d'un validateur p.e.
  `http://www.w3.org/2001/03/webdata/xsv`

Un exemple d'utilisation de DTD : MathML (suite)

```
1  <?xml version="1.0"?>
2  <!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN" "http://www.w3.org/TR/
       MathML2/dtd/mathml2.dtd">
3  <math>
4   <mrow>
5     <msup>
6       <mfenced>
7         <mrow>
8           <mi>a</mi>
9           <mo>+</mo>
10          <mi>b</mi>
11        </mrow>
12      </mfenced>
13      <mn>2</mn>
14    </msup>
15  </mrow>
16 </math>
```

À votre avis, Firefox reconnaitra-t-il maintenant du MathML ?

Vérification dynamique de schéma (typage)



Deux types de vérifcation de schéma (typage)

Deux types de vérification :

Vérification dynamique Un validateur vérifie le type des documents en entrée et éventuellement en sortie de l'application. Au moment de l'exécution.

Vérification statique On vérifie que le programme (l'application) sachant qu'elle prend un certain type (ou un certain schéma) en entrée, renvoie bien le type attendu en sortie. Analyse avant exécution, à la "compilation" (partiel).

Objectifs un peu différents pour un même objectif global : meilleure interopérabilité des applications.

Référence : [**?**]

Et la sémantique dans tout ça ?

```
<!DOCTYPE fiches [
        <!ELEMENT fiches (fiches*)>
          <!ELEMENT fiche (livre, auteur)>
        <!ATTLIST livre titre CDATA #REQUIRED>
        <!ELEMENT auteur (#PCDATA)>
]>
<fiches>
<fiche>
 <livre titre=''XML'' ISBN=''123456''/>
 <auteur> Martin Smith </auteur>
</fiche>
<fiche>
 <livre titre=''RDF(S)'' ISBN=''654321'' />
 <auteur> Marilyn Berton </auteur>
</fiche>
</fiches>
```

Quelle sémantique associer à ce document ?

"Type ou no type ?"

En base de données, on définit le type des données (p.e. un schéma relationnel) puis une instance de ce type (p.e. une base de données relationelles). Pour les données données semi-tructurées (et XML), les données peuvent exister avec ou sans type.

Les types de sont pas oubliés. Ils ne sont juste pas obligatoires. Les définir demande du temps et des efforts. Dans certains cas, le typage n'est pas souhaité (ou pas possible) :

- les données sont irrégulières et la même information peut être représentée sous différentes formes (p.e. les données sont issues de différentes sources) ;
- la structure des données est inconnue (p.e. si les données sont issues d'une source nouvellement découverte) ;
- une partie des données est non typée (plein texte).

Référence : [**?**]

Valider ou non ?

De plus. . .

- Valider peut être coûteux et complexe (la taille prise par la définition du schéma peut être comparable à la taille prise par les données)
- Négociation :

coût en temps et espace
vs
assurance de conformité au schéma (qualité)

Document ou données ?

Distinction courante :

- fichier XML orienté document : article, news, ouvrage, etc
- fichier XML orientés données : typage fort, stockage d'informations ayant une structure commune, modélisable par un schéma

Une question "d'appréciation", il n'existe pas de définition exacte.

Document ou données ? (suite)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <diagrammesUML>
3    <diagramme UML1='yes' UML2='yes'>
4      <name>Diagramme de classes</name>
5      <description>Modélise les objets du système étudié et leurs relations.</description>
6      <objets>
7        Acteurs, cas d'utilisation
8      </objets>
9    </diagramme>
10   <diagramme UML1='yes' UML2='yes'>
11     <name>Diagramme de cas dutilisation</name>
12     <description>Modélise l'ensemble des actions réalisées par le système étudié d'un utilisateur
           extérieur au système</description>
13     <objets>Classes, associations</objets>
14   </diagramme>
15   <diagramme UML1='yes' UML2='yes'>
16     <name>Diagramme de séquence</name>
17     <description>Modélise des interactions entre éléments du système</description>
18     <objets>Rôle, ligne de vie, messages.</objets>
```

Listing 27 – Orienté données

Document ou données ? (suite)

Chile's big heart has buried its history of dictatorship

Eva Salinas
Last updated October 14 2010 12:01AM

The first of the 33 miners to reach the surface yesterday gave a long embrace to the President. Sebastián Piñera, who had arrived hours earlier, beamed from ear to ear.

He is acutely aware that the world is watching Chile as never before — and that over the past two months the Government has turned a large-scale and seemingly preventable mining accident into the best publicity campaign that any country could ask for.

When 2010 began, it looked like the conservative billionaire's biggest challenge as a new president would be to fight off criticism that his supporters were Pinochetistas — defenders of the military coup — and over possible conflicts of interest with his company stocks.

Instead, the year has been full of surprises, with the miners' unprecedented rescue — its engineering innovation, international co-operation and sympathy for the common worker — a crowning achievement.

President Pinera watches the first dry run of the descent of the unmanned Fenix 2 rescue capsule: his approval ratings have shot up

Images AFP/Getty Images

Recommend (0)
Print
Email
Share
J'aime 1

```
<div class="tto-counter">1 of 1</div>
<span class="f-caption">President Pinera watches the first dry run of
<span class="f-credit">Hugo Infante/AFP/Getty Images</span>
</div>
</li>
</ul>
<div class="tto-slideshow-controls">
<a href="#" class="tto-slide-prev" title="Previous"><span></span></a>
<a href="#" class="tto-slide-next" title="Next"><span></span></a>
</div>
</div>
</div>
<div class="byline-timestamp">
<div class="byline special">
<strong class="f-author">Eva Salinas</strong>
<span class="title"></span>
</div>
<div class="f-regular-update">
Last updated October 14 2010 12:01<span class='dateampm'>AM</span>
</div>
</div>
<div id="bodycopy">
<div class="contentpage currentpage" id="page-1">
<p>
The first of the 33 miners to reach the surface yesterday gave a long
to the President. Sebastián Piñera, who had arrived hours earlier, bea
from ear to ear.
</p><p>
He is acutely aware that the world is watching Chile as never before -
that over the past two months the Government has turned a large-scale
seemingly preventable mining accident into the best publicity campaign
any country could ask for.
</p><p>
When 2010 began, it looked like the conservative billionaire's biggest
challenge as a new president would be to fight off criticism that his
supporters were Pinochetistas - defenders of the military coup - and o
possible conflicts of interest with his company stocks.
</p><p>
Instead, the year has been full of surprises, with the miners' unprece
rescue - its engineering innovation, international co-operation and sy
for the common worker - a crowning achievement.
</p><p>
The Government believes that this rescue will be the final piece that
transforms Chile's international image. Instead of being known for its
history of dictatorship and human rights violations, a reputation that
lingered despite 20 years of democracy, Chile will instead be renowned
its wealth, its minerals, its organisation and effectiveness, and its
```

FIGURE 16 – Orienté document

le cnam

Plan du cours

Ce cours JSON

La majeure partie de ce cours JSON est fondée sur le contenu de [**?**] :

Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web Data Management and Distribution*. Cambridge University Press, 2011.

`http://webdam.inria.fr/Jorge/`

- JavaScript Object Notation ;
- Initialement créé pour la sérialisation et l'échange d'objets JavaScript ;
- Langage pour l'échange de données semi-structurées (et éventuellement structurées) ;
- Format texte indépendant du language de programmation utilisé pour le manipuler.

Test

Est-ce que cela ne vous rappelle pas un autre langage ?

Utilisation première : échange de données dans un environnement Web (par exemple applications Ajax [**?**])
Extension : sérialisation et stockage de données

Voir [**?**]

Les bases de JSON

Paire *clef-valeur* (*key-value*)

```
"title": "The Social network"
```

Types atomiques de données : chaînes de caractères (entourées par les classiques guillemets anglais (droits)), nombres (entiers, flottants) et valeurs booléennes (`true` ou `false`).

```
"year": 2010
```

Les bases de JSON (suite)

Un *objet* est un ensemble de paires clef-valeur.

Au sein d'un ensemble de paires, une clef apparait au plus une fois (NB : les types de valeurs peuvent être distincts).

```
{"last_name": "Fincher", "first_name": "David"}
```

Un objet peut être utilisé comme valeur (dite *complexe*) dans une paire clef-valeur.

```
"director": {
  "last_name": "Fincher",
  "first_name": "David",
  "birth_date": 1962
}
```

Les bases de JSON (suite)

Un *tableau* (*array*) est une liste de valeurs (dont le type n'est pas forcément le même).

```
"actors": ["Eisenberg", "Mara", "Garfield", "Timberlake"]
```

Les bases de JSON (suite)

Un *document* est un objet. Il peut être défini par des objets et tableaux imbriqués autant de fois que nécessaire.

```
{
  "title": "The Social network",
  "summary": "On a fall night in 2003, Harvard undergrad and computer\n
        programming genius Mark Zuckerberg sits down at his computer\n
        and heatedly begins working on a new idea. (...)",
  "year": 2010,
  "director": {"last_name": "Fincher",
                  "first_name": "David"},
  "actors": [
       {"first_name": "Jesse", "last_name": "Eisenberg"},
       {"first_name": "Rooney", "last_name": "Mara"}
    ]
}
```

Qu'est-ce qu'un objet JSON ?[?]

object :
- {}
- { *members* }

members :
- *pair*
- *pair* , *members*

pair :
- *string* : *value*

array :
- []
- [*elements*]

elements :
- *value*
- *value* , *elements*

value :
- *string*
- *number*
- *object*
- *array*
- true
- false
- null

le cnam

Représenter des données dans le format JSON ou dans le format XML

Voici un document JSON :

```
{
        "title": "The Social network",
        "year": 2010,
        "genre": "drama",
        "summary": "On a fall night in 2003, Harvard undergrad and computer
                  programming genius Mark Zuckerberg sits down at his computer(...)",
        "country": "USA",
        "director": {
                     "last_name": "Fincher",
                     "first_name": "David",
                     "birth_date": 1962
                    },
        "actors": [
           { "first_name": "Jesse", "last_name": "Eisenberg", "birth_date": 1983, "role": "Mark Zuckerberg" },
           { "first_name": "Rooney", "last_name": "Mara", "birth_date": 1985, "role": "Erica Albright" },
           { "first_name": "Andrew", "last_name": "Garfield", "birth_date": 1983, "role": " Eduardo Saverin" },
           { "first_name": "Justin", "last_name": "Timberlake", "birth_date": 1981, "role": "Sean Parker" }
        ]
}
```

Listing 28 – Document JSON

Test

Représentez les données de ce document dans le format XML.

Représenter des données dans le format JSON ou dans le format XML (suite)

Voici un document XML :

```
<?xml version="1.0" >
<biblio subject="XML" xmlns="http://www.biblioAppliNFE204" >
  <book ISBN="9782212090819" lang="fr" subject="applications" >
    <author> <firstname>Jean-Christophe</firstname> <lastname>Bernadac</lastname> </author>
    <author>
      <firstname>François</firstname> <lastname>Knab</lastname>
    </author>
    <title>Construire une application XML</title>
    <publisher> <name>Eyrolles</name> <place>Paris</place> </publisher>
    <datepub>1999</datepub>
  </book>
  <book ISBN="9782212090529" lang="fr" subject="general" >
    <author> <firstname>Alain</firstname> <lastname>Michard</lastname> </author>
    <title>XML, Langage et Applications</title>
    <publisher>
      <name>Eyrolles</name> <place>Paris</place>
    </publisher>
    <datepub>1998</datepub>
  </book>
  <book ISBN="9782840825685" lang="fr" subject="applications" >
    <author> <firstname>William J.</firstname> <lastname>Pardi</lastname> </author>
    <title>XML en Action</title>
    <publisher> <name>Microsoft Press</name> <place>Paris</place> </publisher>
    <datepub>1999</datepub>
  </book>
</biblio>
```

Listing 29 – Extrait d'un simple fichier XML

le cnam

Représenter des données dans le format JSON ou dans le format XML (suite)

Test

Représentez les données de ce document dans le format JSON. Que constatez-vous ?

Représenter des données dans le format JSON ou dans le format XML (suite)

Voici une table contenant des données régulières.

name	firstname	email	office
Rigaux	Philippe	philippe.rigaux@cnam.fr	37.1.41
Travers	Nicolas	nicolas.travers@cnam.fr	37.1.41
Thion	Virginie	virginie.thion@cnam.fr	37.1.40

Test

Représentez les données contenues dans cette table au format JSON. Que constatez-vous ?

JSON et références

```
{
  person: {name: "alan", phone: 3127786, email: "agg@abc.com"},
  person: &314
                {name: {first: "Sara", last: "Smith-Green"},
                phone: 2136877,
                email: "sara@math.xyz.edu",
                spouse: &443},
  person: &443
                { name: "Fred Green",
                  phone: 7786312,
                  Height: 183,
                  spouse: &314 }
}
```

JSON vs. XML

- JSON plus léger et intuitif que XML,
- Facile à parser pour n'importe quel langage de programmation,
- JSON n'a pas (encore) de langage de spécification de schéma associé,
- JSON n'a pas (encore) de langage de requête associé.

JSON et CouchDB

JSON est le format utilisé dans CouchDB, un système de gestion de documents que vous verrez plus loin dans le cours.